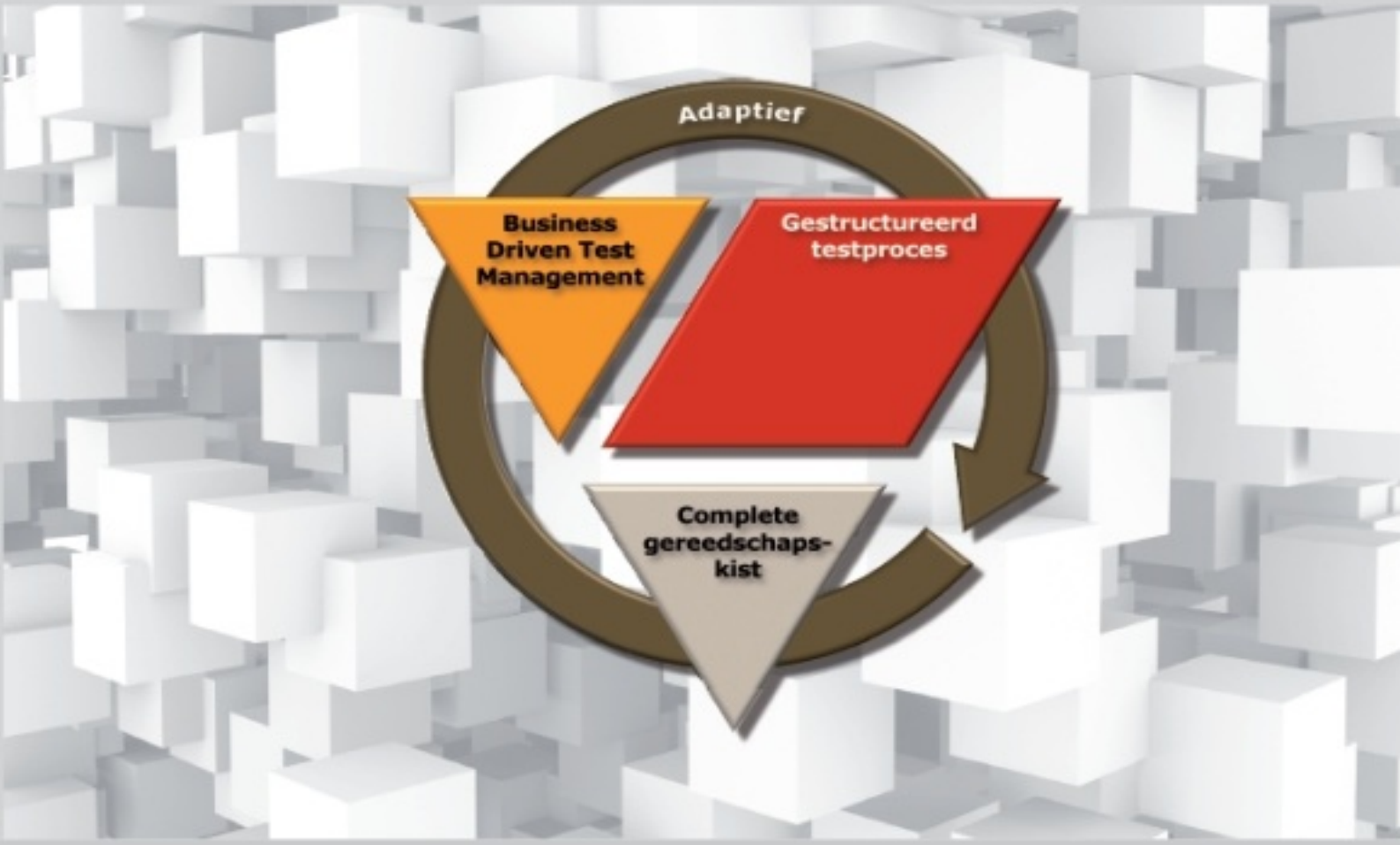
TMap NEXT®

voor resultaatgericht testen



Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon

SOGETI

TMap NEXT®

voor resultaatgericht testen

Tim Koomen
Leo van der Aalst
Bart Broekman
Michiel Vroon

Sogeti Nederland B.V.

2014

ePub-productie LINE UP boek en media bv, Groningen
Omslagontwerp Jan Faber
ISBN 9789075414448 (ePub)

Inhoudsopgave

Voorwoord

Foreword to TMap NEXT, by Rex Black

By the 1980s, Fred Brooks and Barry Boehm had explained that the costs and risks associated with bugs are high and increase as projects progress. In the 1980s, Boris Beizer and Bill Hetzel added another essential insight: testing, done properly, must be influenced by, and influence, risk.

Come the 1990s, leading test professionals were striving mightily to put these insights into action. How do we integrate testing into the entire lifecycle to reduce cost and schedule impacts of bugs? How do we use risk to determine the extent and sequence of testing? How can we report test results in terms of risks mitigated and not mitigated? How can we help project teams make rational decisions regarding the optimal amount of time and energy to expend on testing?

I was one of those test professionals. So were the authors of this book, Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon, and Rob Baarda.

In this book, as in my own books, you can see where that synthesis has taken us. And it has taken us a long way.

As you read this book, comprehensive and comprehensible, pause to remember that it was only ten years ago that test professionals were struggling to understand and implement a complete, consistent approach to testing that managed product quality risks and delivered demonstrable business value. Now, test professionals have a number of fully articulated strategies such as those in this book and my own books which we can use to ensure that we are solving the right testing problems.

In this book, you'll find practical ideas for business driven test management and product risk analysis. These two concepts are so central to testing, yet so often ignored and misunderstood. How does testing serve the needs of the organization and project? What are the risks associated with the system under test that can and should be mitigated through testing? These may seem like obvious questions, yet all too often test teams don't know the answers, or, worse yet, they "know" the *wrong answers*. Getting the right answers to these questions is foundational to a good test effort, and serves to set the direction.

As essential as this direction-setting, foundation-laying material is, there is a lot more to this book than just a means to get the starting point right. Indeed, the authors promise a "full description of the total test process," an explanation of the entire TMap method, from start to finish, and they deliver.

Let me mention a few high points.

The chapter summarizing test design techniques is alone a good reason to have this book close at hand if you work as a test engineer. The discussion on bug management is a fine one. There is a wide-ranging discussion of estimation techniques. There is an intelligent discussion of test metrics, one that doesn't make the all-too-frequent mistake of starting with fancy Excel graphs and tables, but rather by discussing what kind of information that management needs from us as testers.

Pick a random spot in this book and you'll find something interesting.

Whether you are in the first days or the final days of a test effort, you'll find pertinent ideas in this book. Like my own books, this book was obviously written to sit on your desk, ready to serve as a helpful guide on a regular basis, not to collect dust on your bookshelf.

I have in the past complained that as testers we have not built on the foundations of our profession as well as our brothers and sisters in programming. In fact, this has been a key driver for me as an author of testing books. Similarly, Tim, Leo, Bart, Michiel, and Rob have done a fine job of summarizing in this book a number of foundational concepts in software testing. For that reason, I recommend that this book join the set of ready references available to you as together we practice and improve our profession of software testing.

Rex Black, October 2006
Author of *Managing the Testing Process, Critical Testing Processes, Foundations of Software Testing,* and *Pragmatic Software Testing*

President of Rex Black Consulting Services, Inc.
President of the International Software Testing Qualifications Board

Foreword by Luc-François Salvador

It is a great privilege for me to recommend TMap NEXT to you as a member of the continued growing group of people who are professionally involved in improving the quality of a wide range of business processes.

Test Management Approach (TMap) was published in 1996 as a revolutionary management approach for structured testing. Within a few years, TMap was adopted worldwide by companies searching for a structured way to improve their information systems. The objective is to preserve their business processes and market image from damage.

Today TMap has become the de facto world standard for structured testing.

In the meantime, information technology is developing with an astonishing speed. The complexity of chains of information systems supporting the business is increasing rapidly. Defect tolerance is decreasing at the same speed. Experience gained by practice in the area of software testing and the knowledge of IT issues acquired over the years in professional testing are brought together in this complete update: TMap NEXT.

This impressive book is an absolute **must** for the modern business manager, IT manager, and Testing expert. Not only will TMap NEXT inform you as a manager of how to take advantage of the latest developments in the testing profession, but also, even more importantly, assist you in improving the quality of IT incorporated in your current business processes.

Without hesitation I can say this to the reader – if you practice what you read in this book the dream of predictable software engineering to realize business value is indeed attainable by your organization.

Luc-François Salvador
CEO Sogeti SA
Paris

Voorwoord van de auteurs

Een geheel vernieuwd TMap!

Tien jaar na het eerste (Nederlandse) boek, vijf jaar na de tweede druk en drie jaar na de Engelse vertaling zijn we eind 2005 gestart met een grote vernieuwing van de methode, met als resultaat het boek dat nu voor u ligt.

Er is een aantal redenen te geven waarom we deze volledig nieuwe versie uitbrengen:

- De tijd heeft niet stil gestaan. Een groot aantal mensen vroeg daarom om actualisering van de methode, velen hebben hiervoor ideeën aangedragen.
- Dát testen nodig is, is in 2006 voor de meeste organisaties wel duidelijk, de discussie gaat veel meer over het hoe lang, hoe veel en wat getest moet worden.
- In de vorige versie was testen als een op zichzelf staand proces bij de (waterval) nieuwbouw van informatiesystemen beschreven. De huidige ontwikkelingen in de IT gaan veel breder: meer onderhoud dan nieuwbouw, veel pakketimplementaties en iteratieve en agile systeemontwikkeling. Hoewel de methode met de praktijk mee geëvolueerd is, gold dit niet voor het boek. In deze nieuwe versie wordt testen veel meer als een integraal onderdeel van het grotere geheel gezien.
- In veel organisaties is testen in de lijn ingericht, in plaats van puur als projectmatige activiteit. Er zijn diverse lijnorganisatievormen mogelijk, tot complete testfabrieken toe, elk met bepaalde voor- en nadelen. In de literatuur is hier nog weinig aandacht voor.
- En misschien wel het belangrijkst: testen moet veel meer als economische activiteit binnen de IT worden gezien. Tijd en kosten, maar ook de opbrengsten, moeten aan de opdrachtgever duidelijk gemaakt worden. Met deze informatie kan hij/zij het testen sturen op de benodigde hoeveelheid tijd en kosten ten opzichte van de baten: inzicht in kwaliteit en risico's, vertrouwen in het product en project(stuur)informatie. Dit onderdeel van TMap heet BDTM, Business Driven Test Management, en is de rode draad door de methode.

Bij het vernieuwen van de methode hebben we te maken gehad met een aantal uitdagingen. De vele wensen tot

vernieuwing stonden soms haaks op het besef dat veel organisaties al naar volle tevredenheid werken volgens TMap en niet erg enthousiast zijn om de werkwijze overhoop te gooien. We hebben er daarom voor gekozen de hoofdbestanddelen van TMap intact te laten, maar in de details de benodigde vernieuwing te verwerken. Hierbij zijn we zo zorgvuldig mogelijk tewerk gegaan. Dit betekent voor u als lezer een stuk herkenning. Met name de hoofdlijnen van de procesbeschrijvingen aan de hand van fasen, met per fase diverse activiteiten, en de aandacht voor technieken, organisatie en infrastructuur zijn blijven bestaan. Hier bovenop hebben we diverse vernieuwingen aangebracht. De belangrijkste zijn:

- BDTM is als rode draad door het proces heen gevlochten om de opdrachtgever zoveel mogelijk stuurmogelijkheden voor het testen te bieden.
- Het uitvoeren van een productrisicoanalyse is uitgebreid beschreven.
- Diverse begrotingstechnieken voor het testen zijn opgenomen.
- De beheeractiviteit is flink uitgebreid.
- Het inrichten en beheren van de testinfrastructuur is een aparte fase geworden, met nieuw de rol van testinfrastructuurcoördinator.
- De beschrijving van de testontwerptechnieken is flink verbeterd en geactualiseerd. De technieken worden nu gerelateerd aan diverse dekkingsvormen.
- Diverse ondersteunende processen, zoals een permanente testorganisatie, maar ook het selecteren en implementeren van tools en beheren van testomgevingen worden behandeld.
- Testvormen voor het testen van regressie, usability, performance, portabiliteit en beveiliging zijn toegevoegd.
- De methode is veel breder beschreven dan enkel nieuwbouw in een watervaltraject.
- Het gehele boek is verrijkt met meer dan 400 tips, uitdiepingen en praktijkvoorbeelden.

Het nieuwe TMap vatten we daarmee samen in vier essenties:

- Het is gebaseerd op een business driven test management aanpak, die de opdrachtgever in staat stelt het testproces te sturen op rationele en economische gronden.
- Het geeft een volledige beschrijving van het totale testproces.
- TMap bevat een complete ‘gereedschapskist’, dat wil zeggen techniekbeschrijvingen, checklists, procedures, en dergelijke.
- Het is een adaptieve methode, flexibel toe te passen en daarmee geschikt voor alle testsituaties in de meeste ontwikkelomgevingen, zoals nieuwbouw, onderhoud, waterval / iteratief / agile ontwikkeling, maatwerk of pakketsoftware.

TMap biedt de tester en testmanager een leidraad om binnen zijn/haar context resultaat te leveren voor de opdrachtgever.

Omdat de vragen om vernieuwing zowel uit Nederlandstalige landen als uit andere landen komen, wordt het boek vrijwel parallel in zowel het Engels als Nederlands uitgebracht.

Deze nieuwe uitgave is tot stand gekomen zonder betrokkenheid van de oorspronkelijke schrijvers Martin Pol, Ruud Teunissen en Erik van Veenendaal. Hun pionierswerk bij het neerzetten van een complete testmethode wordt door ons nog steeds hoog gewaardeerd. Zonder hen zou TMap niet geweest zijn wat het nu is.

Verder mag duidelijk zijn dat een grote groep mensen geholpen heeft bij de totstandkoming van dit boek. Hun bijdragen varieerden van het inbrengen van ideeën, het delen van ervaringen en ervaringsproducten, het reviewen van hoofdstukken tot het invullen van de benodigde randvoorwaarden. Wat voor bijdrage ze ook geleverd hebben, het is essentieel geweest voor de huidige bereikte kwaliteit. Wij willen hen vanaf deze plaats zeer hartelijk danken. Op gevaar af iemand over te slaan, willen we toch al deze personen met naam noemen.

Als eerste worden de externe reviewers bedankt die belangeloos hun tijd, aandacht en grote kennis hebben gegeven aan het reviewen. Dit waren:
Jan Blaas (KPN)
Jan Boerman (ICTRO, Ministerie van Justitie)
Jarmila Bökkerink-Scheerova (Philips)
Heidi Driessen (Rabobank)
Bart Dooms (Acerta, België)
Ed van der Geest (SNS Bank)
Erwin Kleinveld (Rabobank)
Sander Koopman (Fortis Bank)
Ine Lutterman (Interpay)

Marco Maggi (Ministerie van LNV)
Benjamin Makkes van der Deijl (PGGM)
Jan Mellema (CIP Politie)
Remco Möhringer (Reaal Verzekeringen)
Paul van der Molen (Cordares)
Brian Taylor (FortisBank)
Ubel van Tergouw (UWV)
Sylvia Verschueren (ABN AMRO)
Hans Vedder (Friesland Bank)
Dennis van Velzen (AFAS ERP Software BV)

Ook de grote groep Sogeti collega's heeft natuurlijk enorm geholpen. Hun bijdragen varieerden van het lid zijn van de klankbordgroep, reviewen tot bijdragen van ideeën en zelfs complete teksten:
Richard Ammerlaan, Luciëlle de Bakker, Eric Begeer, Paul Bentvelzen, John Bloedjes, Martin Blokpoel, Hester Blom, Martin Boomgaardt, Raoul Gisbers, Frank Goorhuis, Guy Holtus, Bart Hooft van Iddekinge, André Huikeshoven, Marco Jansen van Doorn, Ralph Klomp, Rob Kuijt als gastschrijver van hoofdstuk 7, Peter van Lint, Dominic Maes (Sogeti België), Willem-Jan van der Meer, Henk Meeuwisse, Gerrit Mudde, Guido Nelissen, Bert Noorman, André van Pelt, Elisabeth Reitsma, Ewald Roodenrijs, Rob Smit, Gert Stad, Marc Valkier, Thomas Veltman, Johan Vink, Ben Visser, Harm de Vries, expertisegroep voor usability (Kinga Visser, Gina Utama, Ronald Oomen, Wolter van Popta, Robin Klein, Thomas Veltman, Mans Scholten, Mark Schut, Martien Adema, Jeroen Bultje).

Dit boek is niet mogelijk geweest zonder de steun van het managementteam van Software Control. In het bijzonder willen we Wim van Uden en Mark Paap bedanken. Hun waardering en belangstelling zijn stimulerend geweest voor ons.

De afgelopen periode hebben wij met veel plezier aan het boek gewerkt. Of u nu een ervaren TMap-gebruiker bent of een onervaren tester, wij hopen u met dit boek een helder verhaal te bieden over hoe het testproces in al zijn aspecten het best ingericht kan worden en u te inspireren met de vele ideeën, uitdiepingen en tips. We zijn trots op het resultaat en hopen dat u dat als lezer zult beamen.

Tim Koomen
Leo van der Aalst
Bart Broekman
Michiel Vroon
Rob Baarda (projectleiding en redactie)

Rotterdam, oktober 2006

Aanbevelingen

"Voor onze bedrijfsvoering is het essentieel dat de onderliggende ICT van goede kwaliteit is. Nieuwe opleveringen mogen de bedrijfsprocessen niet verstoren en moeten in één keer goed zijn. Om deze kwaliteit aan te tonen en de business voldoende zekerheid te geven zijn goede en gestructureerde testprocessen erg belangrijk. Dit vereist een goede samenwerking tussen business en ICT. Ik onderstreep dan ook volledig het belang van Business Driven Test Management, zoals verwoord in dit boek en zoals wij dit stap voor stap implementeren. De TMap methode is een belangrijk instrument voor onze testprocessen en wij zijn blij dat wij als Fortis een bijdrage hebben kunnen leveren aan deze nieuwe versie."
Gert Heslenfeld
Sectormanager IS / Accountmanager Information Services
Fortis Verzekeringen

"Het ministerie van Landbouw, Natuur en Voedselkwaliteit (LNV) geeft jaarlijks veel geld uit aan beheer, pakketimplementaties en (ver)nieuwbouw van haar software. Standaardisatie bij alle disciplines in de softwarelifecycle is noodzakelijk voor de control daarop. Testen is voor die lifecycle steeds belangrijker vanuit kwaliteits- en kostenperspectief. Juist daarom heeft het ministerie gekozen voor een standaard methodologie en een permanente testorganisatie. En dan is het goed om vast te stellen dat deze standaard TMap evolueert en zich niet alleen aanpast aan de snelle technische ontwikkelingen maar o.a. met BDTM ook inspeelt op de eisen van opdrachtgevers."
Pieter C.A. Arends
Manager Project management, Consultancy & Test services
MT-lid Dienst ICT-Uitvoering
Ministerie van Landbouw, Natuur en Voedselkwaliteit

"TMap is één van de belangrijke instrumenten om de kwaliteit van onze IT producten te borgen. REAAL Verzekeringen heeft daarom met plezier in TMap nieuw een bijdrage geleverd aan het deel over ontwikkeltesten."
Nico Jongerius
Lid Hoofddirectie / directeur ITF REAAL Verzekeringen

"TMap NEXT, is definitely a must read for test organizations using TPI Model for Test Process Improvement and/or TMap approach to testing. At the same time, the book is not dependent on understanding of these models, and is a useful resource for organizations implementing risk based test strategy in their organizations.. The process, techniques and tools described in this book will enable a test professional to better balance the test cost with the benefits provided by testing, thus making it easier to get management buy in for the test project."
Ruku Tekchandani
SW Validation CoP Program Manager
Intel Corporation, United States of America

"In many organisations all over the world TMap is the standard for testing for many years. The experiences gained using TMap over all these years and the new approaches in software development are the motivators to update TMap and to write this book. For instance, product risk analysis and business driven test management are added to TMap as well as use case testing and exploratory testing. The "new" TMap standard is very useful to solve the testing problems of today and tomorrow."
Prof. Dr. Andreas Spillner
University of Applied Sciences Bremen
Member of the German Testing Board (SIGIST)
Germany

"This is testing in a book. If you need to implement the practice of testing, this is a tool that is essential. Tim Koomen and colleagues have taken a great and popular book, TMap, and have improved it in its updating – TMap NEXT highlights testing as an economic activity within IT and, the new emphasis on Business Driven Test Management empowers the business to get the economic and other benefits from IT that they have only dreamed of until now. This book will become a classic on software testing."
Wayne Mallinson and Peter Sage
Communications and Technical Directors respectively – Test and Data Services (Pty) Ltd Founder and current Chairman respectively – CSSA Special Interest Group in Software Testing (SIGiST)
Gauteng South Africa

"TMap contains practical, proven ideas and methods for a risk based testing approach. The updated TMap is an important read for anyone endeavoring to understand how business driven test management fits within the context of the end-to-end process of business driven development."
Dr. Daniel Sabbah
General Manager, IBM Rational Software
United States of America

"I like this book because of it being a detailed recipe for how to organize, plan, execute and control testing. Most techniques are explained in detail, with lots of examples. Especially useful are the chapters about handling risk, and the one about test case design. The latter one is great for system and acceptance testing, while most other literature concentrates on lower level testing."
Hans Schaefer
Independent test consultant
Leader of the Norwegian Testing Board (SIGIST)

"TMap is for testing what ITIL is for operational processes. This book is a comprehensive yet highly accessible guide for both test managers and test professionals. It is truly a meaningful help to anyone who's involved in the testing business; whether you have one or more specific questions, or if you are in search of a whole testing framework or well targeted process improvements."
Wim Waterinckx
Process manager testing
KBC Group
Belgium

"TMap is the most thorough to help organise testing in large IT companies."
Dorothy Graham, Grove Consultants
Great Britain

"I've been a heavy user of TMap since the availability of the book in English. I have utilized TMap for testing process communication, competence development and expanding the 'skill-toolbox' of my testers in Nokia. The basic framework is very similar, but it is now much more useful because of the explicit support for wider selection of contexts.
I like the clarity of the big picture of test design: how test planning includes risk-based test strategy that links to test types, and how eventually various test design techniques provide the right depth for testing at expected coverage. As a bonus the framework suits well to expanding with your own techniques and test types."
Erkki Pöyhönen
TietoEnator Quality Assurance Competence Center, previously with Nokia
Finland

"As the quality assurance was moving from a project issue to an issue for the company board our test organisation needed to unify and structure the test process. TMap NEXT follows the projects phases and has as a logical toolbox. This helped us to complement our existing test process to meet the company board's timescale. TMap NEXT also gave us useful information how to handle people, tools, infrastructure and the organisation that in many ways has influenced the organisation of the test centre at the Swedish Tax Agency."
Henrik Rylander
Head of Test Centre
Test Unit
The Swedish Tax Agency
Sweden

1 Inleiding

TMap is de Test Management approach van Sogeti, dé aanpak voor het gestructureerd testen van informatiesystemen. De aanpak is universeel beschreven omdat de specifieke invulling van dé testaanpak afhankelijk is van de situatie waarin deze wordt toegepast. TMap bevat dan ook vele handvatten om in uiteenlopende situaties de universele aanpak concreet in te vullen.

TMap is in een viertal essenties samen te vatten:

1. TMap is gebaseerd op een business driven testmanagement aanpak.
2. TMap beschrijft een gestructureerd testproces.
3. TMap bevat een complete gereedschapkist.
4. TMap is een adaptieve testmethode.

De eerstgenoemde essentie is direct te relateren aan het feit dat de business case van IT (de rechtvaardiging van een project) steeds belangrijker wordt voor organisaties.

Voor testen betekent dit: de keuzes voor het afdekken van bepaalde risico's, het te bereiken resultaat en de hoeveelheid te besteden geld en tijd moeten gebaseerd zijn op rationele en economische afwegingen. Om hier in TMap invulling aan te geven, is de business driven testmanagement aanpak ontwikkeld, welke kan worden gezien als de 'rode draad' van het gestructureerde TMap testproces.

TMap geeft door de beschrijving van een gestructureerd testproces (essentie 2) en door het aanreiken van een complete gereedschapkist antwoord op de klassieke vragen: wat/wanneer, hoe, waarmee en wie. Bij de beschrijving van het testproces wordt gebruik gemaakt van het TMap faseringsmodel: een aan de ontwikkelingscyclus gerelateerde beschrijving van de testcyclus. Het faseringsmodel beschrijft *wat/wanneer* moet worden uitgevoerd.

Daarnaast moeten, voor het goed kunnen uitvoeren van het testproces, diverse zaken op het gebied van infrastructuur (*waarmee*), technieken (*hoe*) en organisatie (*wie*) worden ingericht. TMap geeft hierover veel praktisch toepasbare informatie in de vorm van onder andere voorbeelden, checklists, techniekbeschrijvingen, procedures, testorganisatiestructuren en testomgevingen/-tools (essentie 3).

Verder is TMap flexibel opgezet zodat het toepasbaar is in diverse systeemontwikkelsituaties: bij zowel nieuwbouw als onderhoud van een systeem, bij een zelfontwikkeld systeem of aangeschaft pakket en bij uitbesteding van (delen van) het testen (essentie 4).

In dit hoofdstuk wordt achtereenvolgens een overzicht gegeven van: hoe TMap dé marktstandaard is geworden, waarom TMap is vernieuwd en wat de kernpunten van TMap zijn. Ook wordt een aantal suggesties gedaan met betrekking tot welke hoofdstukken voor welke doelgroepen het meest interessant zijn om te lezen.

1.1 Geschiedenis van TMap

In de Nederlandse en Belgische testwereld is TMap een bekend begrip. Deze testaanpak bestaat dan ook al enige tijd. Hoewel het niet nodig is om de geschiedenis van TMap te kennen om de testaanpak te kunnen begrijpen of toe te kunnen passen, lichten we in deze paragraaf toch een 'tipje van de sluier' op.

Marktstandaard

Dit boek is voorafgegaan door "Testen volgens TMap" in 1995 en de tweede druk daarvan in 1999 (Pol, Teunissen en Van Veenendaal). Het bleek en blijkt nog steeds een bestseller te zijn. De vraag naar het boek is enorm. TMap heeft zich in de afgelopen jaren ontwikkeld tot dé standaard voor het testen van informatiesystemen. Het wordt momenteel toegepast in honderden bedrijven en instellingen, waaronder de meeste Nederlandse en Belgische banken, verzekeringsmaatschappijen, pensioenfondsen en overheidsinstellingen. Dat TMap als marktstandaard wordt gezien blijkt wel uit feiten zoals, leveranciers van testtools die adverteren met "toepasbaar in combinatie met de TMap-technieken", testprofessionals die ervoor zorgen dat TMap-ervaring op hun CV prijkt of, steeds vaker voorkomend, dat ze TMap-gecertificeerd zijn, personeelsadvertenties waarin testprofessionals met TMap-kennis worden gevraagd en onafhankelijke opleidingsinstituten die TMap-opleidingen geven. Vermeldingswaardig hierbij is nog de publicatie in de "Computable"

van 30 september 2005, waaruit blijkt dat TMap de meest gevraagde competentie is! Dus nog voor bijvoorbeeld Java, ITIL en Unix.

COMPETENTIE TOP 40		
Expertise	**Score**	**Plaats**
TMap	389	1
Java	351	2
ITIL	340	3
Unix	330	4
Windows NT/2000	316	5
Cobol	271	6
XML	243	7
Oracle PL/SQL	240	8
J2EE	226	9
Microsoft .NET	217	10

(Computable 39, 30 september 2005)

Ook internationaal gezien mag TMap zich verheugen in een toenemende belangstelling. Om niet-Nederlandstaligen tegemoet te kunnen komen zijn ook een Duitstalige- en een Engelstalige versie uitgebracht.[1]

Daarnaast neemt het gebruik in hoog tempo toe in andere marktsegmenten, zoals in de embedded software industrie. Aangezien testen in deze industrie verschilt van testen in de administratieve wereld is speciaal daarvoor een ander, op TMap-concepten gebaseerd, boek geschreven [Broekman, 2003].

De kracht van TMap

De kracht van TMap kan voor een groot deel worden toegeschreven aan de vele praktijkervaringen die in de aanpak zijn verwerkt. Deze ervaringen zijn afkomstig van honderden professionele testers in evenzovele projecten uit de afgelopen twintig jaar! Het boek is een waardevol hulpmiddel voor de meeste, zo niet alle, uitdagingen op testgebied van dit moment en in de nabije toekomst. Een nadeel van een boek is dat de inhoud per definitie statisch is, terwijl er in de IT met grote regelmaat weer nieuwe inzichten, systeemontwikkelmethoden, enzovoort ontstaan.

Het zou bedrijfseconomisch onverantwoord zijn om bij elke nieuwe ontwikkeling in de IT een gewijzigde versie van het TMap boek samen te stellen en uit te brengen. Om in te kunnen spelen op actuele ontwikkelingen in de IT, worden er op regelmatige basis toepassingsvarianten[2] geschreven. In een dergelijk document, soms ook ‘white paper’ genoemd, worden de lezer één of meer varianten aangereikt om TMap in een specifieke situatie toe te kunnen passen. Veel van deze papers zijn opgenomen in het boek TMap Test Topics [Koomen, 2004]. Voor het informeren van TMap gebruikers over nieuwe toepassingsvarianten wordt van diverse communicatiemiddelen gebruik gemaakt. Denk hierbij aan het grote aantal presentaties en workshops die gegeven worden op zowel nationale- als internationale testcongressen, aan de drukbezochte TMap Test Topics sessies (waarin actuele testthema’s worden gepresenteerd) en aan de vele publicaties in de diverse vakbladen. Dit alles maakt TMap tot het wat het nu is: “een complete testaanpak, waarmee iedere organisatie elke testuitdaging, nu en in de toekomst, succesvol aan kan gaan!”

Tip

Kijk eens op www.tmap.net. U vindt daar onder andere:

- downloads (o.a. toepassingsvarianten, checklists, testontwerptechnieken en een woordenlijst)
- de TMap Test Topics presentaties
- in de pers verschenen TMap artikelen
- TMap nieuwsbrieven (als u deze automatisch wilt ontvangen kunt u zich op deze site hier ook voor aanmelden).

1.2 TMap evolueert mee

“Waarom dan een nieuwe versie van TMap” zult u zich wellicht afvragen?

“Was er iets mis met de vorige versie?” Nee, er was helemaal niets mis met de vorige versie, maar sinds deze op de markt verscheen, hebben er diverse nieuwe ontwikkelingen op zowel IT- als testgebied plaatsgevonden. En om TMap zo

compleet en actueel als mogelijk te houden zijn deze in de nieuwe versie verwerkt. Het betreft hier ontwikkelingen als het steeds belangrijker worden van IT voor organisaties en een aantal vernieuwingen op systeemontwikkelingsgebied. Hiernaast blijkt de tester bij zijn werkzaamheden meer geholpen te worden door een als leidraad geschreven testaanpak dan door een handboek testen. Een toelichting hierop volgt in de volgende paragrafen.

IT steeds belangrijker voor organisaties

Sinds eind jaren negentig is het gebruik van software of in het algemeen informatietechnologie steeds belangrijker geworden voor organisaties, waardoor IT projecten steeds vaker vanuit een gebruikersorganisatie worden geinitieerd en gestuurd. Dit is ingegeven door de volgende ontwikkelingen:

- Kostenreductie in ontwikkeling en beheer van IT
  IT moet goedkoper en de business case (het waarom in combinatie met kosten en verdiensten) moet steeds duidelijker worden neergezet.
- Groeiende automatisering van bedrijfsprocessen
  Steeds meer bedrijfsprocessen binnen organisaties worden of geautomatiseerd of sterk afhankelijk van andere geautomatiseerde processen.
- Snellere inzet van automatisering
  Door de groeiende automatisering is IT van een ondersteunend bedrijfsmiddel naar een concurrentie onderscheidend bedrijfsmiddel gegroeid. Dit betekent dat de flexibiliteit en snelheid waarmee dit middel kan worden ingezet van levensbelang is voor het winnen van de concurrentieslag.
- Kwaliteit van automatisering wordt belangrijker (zie praktijkvoorbeeld “Gevolgen van falende software”)
  De mondigheid van eindgebruikers van IT in combinatie met het feit dat hoofdbestuurders verantwoordelijk worden gesteld voor de juistheid van deze IT, zorgen voor (hernieuwde) belangstelling voor kwaliteit.

Deze vier aspecten worden ook wel kernachtig samengevat als “meer voor minder en sneller en beter”. Een consequentie hiervan is dat IT projecten steeds vaker dynamischer en hectischer van aard worden. Hierdoor kan grote druk op de testers komen te staan en kan het relatieve aandeel testen binnen de IT toenemen. Om het testproces toch beheersbaar te maken én te houden is voor TMap de ‘ business driven testmanagement’ (BDTM) aanpak ontwikkeld. Deze aanpak is in dit boek geïntegreerd. Hierbij is het maken van een teststrategie direct gerelateerd aan de risico’s. De opdrachtgever kan op deze wijze verantwoorde, op risico’s gebaseerde keuzes maken in het testproces. Door het maken van deze keuzes heeft de opdrachtgever zelf nadrukkelijk invloed op de doorlooptijd en de kosten van het testproces.

Praktijkvoorbeeld

Gevolgen van falende software

Doordat de IT belangrijker is geworden binnen organisaties is de impact van eventuele softwareproblemen vergroot.

Enkele voorbeelden zijn:

- omzetverlies
- imagobeschadiging
- schadeclaims
- productiviteitsverlies.

Dat het hierbij om aanzienlijke verliezen kan gaan blijkt wel uit de volgende ‘realworld’ voorbeelden:

- Een sportartikelen fabrikant leed 24% omzetverlies (€100 miljoen) in één kwartaal ten gevolge van softwarefouten in de voorraadadministratie. In de dagen daarna, nadat het bedrijf bekend had gemaakt dat ze problemen met de software hadden, daalde de waarde van de aandelen met meer dan 20%.
- Door een fout in hun encryptie software, toonde een financieel adviesverlenende organisatie ‘social security numbers’ en wachtwoorden van hun klanten in leesbare tekst op hun website. Dit veroorzaakte grote ophef onder de klanten en leidde tot omvangrijk klantenverlies.
- Nadat een farmaceutische groothandel failliet was gegaan, werd door de moedermaatschappij een schadeclaim van €500 miljoen neergelegd bij de softwareleverancier voor het vermeend falen van de ‘enterprise software’.

Vernieuwingen op systeemontwikkelingsgebied

De testprincipes die aan TMap ten grondslag liggen zijn ontstaan in de jaren tachtig en uiteraard gerelateerd aan systeemontwikkelmethoden uit die tijd. Tegenwoordig wordt er bij het refereren aan deze methoden vaak over watervalmethoden gesproken. Deze bezitten als belangrijkste kenmerk het zuiver sequentieel uitvoeren van de diverse ontwikkelactiviteiten. Daarnaast is er groeiende belangstelling voor andere methoden. De nieuwe generatie systeemontwikkelmethoden bezitten als belangrijkste kenmerk het incrementeel en iteratief ontwikkelen, waarbij het testproces steeds vaker in het ontwikkelproces is geïntegreerd. Denk hierbij aan methoden als Dynamic Systems Development Methodology® (DSDM), Rational Unified Process® (RUP), Rapid Application Development (RAD) en de Agile aanpak.

Aangezien testen tijdens diverse activiteiten een raakvlak heeft met de gekozen systeemontwikkelmethode kan het niet anders dat ook een testaanpak als TMap mee moet evolueren met deze veranderingen op systeemontwikkelingsgebied. Hoe TMap in bepaalde situaties toe te passen is, is vaak al vastgelegd in de eerder genoemde toepassingsvarianten. Vrijwel alle kernpunten uit de toepassingsvarianten zijn nu ook in deze nieuwe versie van TMap geïntegreerd, waardoor het boek weer een actueel en een zo compleet mogelijk overzicht geeft hoe TMap in diverse situaties kan worden toegepast.

Leidraad voor testaanpak

De vorige twee ontwikkelingen, het steeds belangrijker worden van IT binnen organisaties en de vernieuwingen op systeemontwikkelingsgebied, schetsen de toegenomen dynamiek van de diverse ontwikkel- en testomgevingen. In een dergelijke situatie wordt een handboek vaak als te rigide beschouwd. Verder blijkt het testvak in toenemende mate te worden uitgeoefend door een breder publiek met een grote diversiteit aan competenties. Hierdoor is er een grotere behoefte ontstaan aan uitgebreidere beschrijvingen hoe bepaalde TMap activiteiten kunnen worden uitgevoerd. Een consequentie hiervan voor het beschrijven van de testaanpak is dat deze meer als leidraad (‘bij de hand nemend’) dan als handboek zal worden beschreven.

De voor u liggende versie van TMap is geheel herzien in reactie op de geschetste ontwikkelingen én op de vele verzoeken om aanpassingen en uitbreidingen op het boek.

Kortom, deze versie van *TMap biedt de tester een leidraad om resultaat te leveren voor de opdrachtgever.*

1.3 Wat TMap biedt

Maar wat biedt TMap nu, wat kun je er mee? Het antwoord op deze vraag vindt u in deze paragraaf, waarin een globaal overzicht wordt gegeven van de hulp die TMap kan bieden en waar TMap toepasbaar is. In het vervolg van dit boek wordt hier dieper op ingegaan en wordt er uitgebreid aandacht besteed aan de wijze waarop TMap in diverse situaties toegepast kan worden.

1.3.1 Waarbij TMap helpt

Om de tester te helpen bij zijn werk is in TMap voor het uitvoeren van bepaalde activiteiten aangegeven hoe deze uitgevoerd kunnen worden of hoe deze door TMap worden ondersteund. Dit betreft hulp bij:

- het vertalen van de wensen van de organisatie (opdrachtgever) naar een concrete testaanpak en aansturing van de uitvoering;
- het om kunnen gaan met de diverse IT ontwikkelaanpakken door een testmanager, testcoördinator en/of tester, ieder vanuit hun eigen verantwoordelijkheid;
- het uitvoeren van onder andere een;
  - productrisicoanalyse;
  - teststrategie;
  - (niet-)functionele test;
- het organiseren van testers in bijvoorbeeld een staande testorganisatie of in flexibele projectteams; het inrichten en beheren van de testinfrastructuur in het project en over projecten heen; het maken van testontwerpen en het gebruik van diverse testontwerptechnieken; het voorbereiden, specificeren en uitvoeren van de tests door de procesmatige beschrijving van de TMap processen; het uitvoeren van diverse activiteiten, doordat praktijkvoorbeelden, tips en zelfs uitdiepingen van bepaalde onderwerpen zijn toegevoegd; het aangeven van het resultaat van de test, dat wordt vastgelegd in concrete afspraken; het rapporteren van testresultaten vanuit het perspectief van de opdrachtgever; zoveel

mogelijk te redeneren van buiten het testproces naar binnen, bijvoorbeeld door het beantwoorden van de nutsvraag (wat brengt testen eigenlijk op?) en door het gebruik van algemene projectinformatie;
- het kiezen van de juiste testgesprekspartner voor een bepaalde opdrachtgever;
  - een testmanager is in staat om het testen in een breder perspectief te zien (omgevingsgericht) en is daardoor een geschikte gesprekspartner voor opdrachtgevers op managementniveau;
  - een testcoördinator richt zich vooral op het interne testproces en zal daardoor vaak gesprekspartner zijn van bijvoorbeeld projectleiders;
  - een tester is taakgericht en zal zich vanuit deze competentie hoofdzakelijk bezig houden met het ontwerpen en uitvoeren van testgevallen.

1.3.2 Waar TMap toepasbaar is

De IT ontwikkelaanpak kent tegenwoordig een groot aantal varianten. TMap gaat daar actief mee om en onderkent op voorhand de volgende mogelijkheden waar TMap kan worden toegepast:

- bij zowel een demand-supplier verhouding (bijvoorbeeld bij outsourcing) tussen opdrachtgever, ontwikkelaar en tester met ieder eigen verantwoordelijkheden als ook bij gezamenlijke interactieve aanpakken;
- bij zowel iteratieve-, incrementele-, waterval- als agile aanpakken;
- bij nieuwbouw, onderhoud en migratie van informatiesystemen;
- in situaties met combinaties van zelfontwikkeling, hergebruik, gebruik van standaardpakketten en assembleren van ingekochte modules, dit alles onder IT-architectuur;
- bij het, naast de functionele requirements, betrekken van de non-functionele requirements van het informatiesysteem bij de testaanpak;
- bij situaties waarin veel aandacht aan het communicatieproces en bijbehorende vaardigheden besteed moet worden.

Tip

Het implementeren of verbeteren van een testaanpak is niet iets dat je ‘zomaar even’ doet. Het vereist onder andere kennis van de actuele testvolwassenheid van een organisatie en van de omgeving waarin het testen moet worden ingevoerd of verbeterd. Hiernaast moet de noodzaak duidelijk zijn waarom bepaalde testaspecten moeten worden verbeterd. Het blijkt in de praktijk vaak moeilijk te bepalen welke stappen in welke volgorde moeten worden genomen om het testen te implementeren of te verbeteren. Het “Test Process Improvement”-model [Koomen, 2006] is een veelgebruikt model bij het stapsgewijs implementeren en verbeteren van een testaanpak.

1.4 Leeswijzer en de belangrijkste veranderingen

Als u dit boek[3] leest is de kans groot dat u een antwoord wilt hebben op een specifieke (test)vraag. Als u hiervoor niet het hele boek wilt lezen kunnen onderstaande beschrijving van de indeling van het boek en de leeswijzer u wellicht sneller naar een antwoord leiden.

Om de verschillen met de vorige versie te doorzien, moet eigenlijk het hele boek gelezen worden. De meeste teksten zijn herschreven en het boek bevat meer dan 400 nieuwe ideeën, tips en voorbeelden. Toch kunnen wij ons voorstellen dat een ervaren TMap gebruiker zich snel op de hoogte wil stellen van de belangrijkste veranderingen. Voor hen is een paragraaf toegevoegd dat verwijst naar de hoofdstukken, paragrafen en onderwerpen waarin de belangrijkste veranderingen zijn aangebracht (paragraaf 1.4.3).

1.4.1 Indeling boek

Het boek is opgebouwd uit drie delen. Een algemeen, processen en componenten deel.

Het algemene deel beschrijft het kader en belang van testen. Verder wordt, naast het geven van de geschiedenis en de evolutie van TMap, beschreven wat TMap biedt en een beknopt overzicht van TMap gegeven.

Het processen deel beschrijft de testprocessen mastertestplan (managen van het totale testproces), acceptatie- en systeemtesten en ontwikkeltesten. Verder worden in dit deel daaraan ondersteunende processen (o.a. permanente

testorganisatie en inrichten en beheer testomgevingen) beschreven.

Het componenten deel beschrijft een aantal componenten die zelfstandig (o.a. kwaliteitsattributen en testrollen) of als hulp bij de processen kunnen worden gebruikt (o.a. productrisicoanalyse en testontwerptechnieken). Voor een compleet overzicht van deze componenten wordt verwezen naar tabel 1.1 “Leessuggestie” in de volgende paragraaf.

In het boek zijn op diverse plaatsen definities, praktijkvoorbeelden, tips en uitdiepingen beschreven. Deze zijn herkenbaar aan de titel en een kader of een grijze achtergrond.

1.4.2 Leeswijzer

De primaire doelgroep van dit boek wordt gevormd door degenen die direct bij het testproces zijn betrokken. Een testteam kan het boek hanteren als leidraad voor de uitvoering van de testactiviteiten. Voor hen die wat verder van het primaire testproces af opereren, zoals opdrachtgevers, eindgebruikers en EDP-auditors, biedt het boek een goed inzicht in het vakgebied testen. Hiertoe zijn enkele hoofdstukken opgenomen over de achtergronden en de inrichting van het testen.

“TMap NEXT” is geen leesboek dat van voor naar achter hoeft te worden gelezen. Afhankelijk van de betrokkenheid bij het testen zullen hoofdstukken grondig, vluchtig of zelfs niet worden gelezen. De interesse voor de diverse hoofdstukken zal per doelgroep verschillen. Als hulp bij het kiezen van welke hoofdstukken interessant zijn voor bepaalde vragen of doelgroepen is er een leessuggestie opgenomen (tabel 1.1). Als mogelijke vragen en doelgroepen worden hierin onderscheiden:

A. U wordt gevraagd een test op te zetten en de uitvoering daarvan te gaan leiden. (testmanagers, testcoördinatoren en dergelijke)
B. U wordt gevraagd een applicatie te gaan testen. (testers, ontwikkelaars, gebruikers en beheerders)
C. U wordt gevraagd de kwaliteit te beoordelen van een test met bijbehorende processen en producten. (EDP-auditors en kwaliteitszorgmedewerkers)
D. U bent verantwoordelijk voor de IT-afdeling of eigenaar van een applicatie en wilt dat er professioneel getest wordt. (opdrachtgevers voor ontwikkelings- en testprocessen en betrokken lijnmanagement)
F. Vanuit uw opleiding bent u geïnteresseerd in het vakgebied testen. (studenten informatica/bedrijfseconomie en docenten)
F. U bent verantwoordelijk voor HRM binnen de organisatie en u krijgt testers in dienst. (medewerkers personeel & organisatie).

Hoofdstuk		A	B	C	D	E	F
Algemeen							
H01	Inleiding	√	√	√	√	√	√
H02	Kader en belang van testen	√	√	√	√	√	√
H03	TMap in essenties	√	√	√	√	√	√
Processen							
H04	Inleiding processen	√	√	√	√	√	
H05	Mastertestplan, managen van het totale testproces	√		√	√	√	
H06	Acceptatie- en systeemtesten	√	√	√	√	√	
H07	Ontwikkeltesten	√	√	√			
H08	Ondersteunende processen	√			√		√
Componenten							
H09	Productrisicoanalyse	√					
H10	Kwaliteitsattributen en testvormen	√	√				
H11	Begrotingstechnieken	√					
H12	Bevindingenbeheer	√	√				
H13	Metrics	√					
H14	Testontwerptechnieken		√			√	
H15	Toetstechnieken	√	√				
H16	Testrollen	√					√

Toelichting tabel: √, zinvol voor desbetreffende doelgroep om hoofdstuk grondig te bestuderen. De overige hoofdstukken kunnen optioneel worden bestudeerd (dit geldt zeker voor groep E).

Tabel 1.1: Leessuggestie

1.4.3 De belangrijkste veranderingen

De veranderingen in TMap zijn met alle hoofdstukken van het boek geïntegreerd. Het is daarom onmogelijk om precies *die* plaatsen in het boek aan te geven, waar de veranderingen zijn aangebracht. Wij denken echter dat een ervaren TMap gebruiker, ook zonder het lezen van het gehele boek, de belangrijkste doorgevoerde veranderingen kan doorgronden. Voor hen is onderstaande lijst opgesteld met daarin de verwijzingen naar de hoofdstukken, paragrafen en onderwerpen waarin de belangrijkste veranderingen zijn aangebracht:

- TMap is verdeeld in vier essenties. Hoofdstuk 3 beschrijft dit uitgebreid.
  Hierin bevindt zich ook een overview van business driven test management.
- Het uitvoeren van een productrisicoanalyse is uitgebreid beschreven in hoofdstuk 9.
- Business driven test management en de productrisicoanalyse hebben grote invloed op onder andere opdrachtoriëntatie, strategie, begroting en planning. Zie hiervoor de hoofdstukken 5 en 6.
- De fase Beheer van het totale testproces en de fase Beheer van de acceptatie- en systeemtesten zijn aparte fasen geworden (gescheiden van de fase Planning). Zie hiervoor de paragrafen 5.3 en 6.3.
- De fase Inrichting en beheer infrastructuur is nieuw. Zie hiervoor paragraaf 6.4.
- De fase Planning en de fase Beheer (van het totale testproces) beschrijven nu vooral de testmanagementactiviteiten. Voor het zien van alle verschuivingen kan het beste de activiteitenschema's uit de hoofdstukken 5 en 6 worden doorgenomen.
- In hoofdstuk 11 zijn een groot aantal nieuwe begrotingstechnieken opgenomen.
- Een nieuwe 'blik' op ontwikkeltesten is in hoofdstuk 7 beschreven.
- Naast de testproces hoofdstukken is er een nieuw hoofdstuk opgenomen met ondersteunende processen. Dit hoofdstuk 8 gaat in op onderwerpen als: testbeleid, permanente testorganisatie, testomgevingen, testtools en de testprofessional.
- In hoofdstuk 14 is, vooral in de eerste drie paragrafen, een nieuwe invalshoek te vinden over dekking, basistechnieken en testontwerptechnieken. De testontwerptechnieken beslistabeltest en datacombinatietest zijn gewijzigd. Nieuwe technieken zijn use case test en exploratory testing. Naast deze vier zijn nog zeven andere technieken in paragraaf 14.4 beschreven.
- In paragraaf 10.3 is de set van testvormen uitgebreid met testen van regressie, usability, performance, portabiliteit en beveiliging.

Noten

1 Actuele referenties zijn te vinden op www.tmap.net.

2 Recente publicaties zijn te vinden op www.tmap.net.

3 Omwille van de leesbaarheid van het boek is er voor gekozen om geen hij/zij vermeldingen op te nemen. Hoewel er in het boek alleen 'hij' vermeldingen zijn opgenomen wordt hiermee uitdrukkelijk ook zij bedoeld.

2 Kader en belang van testen

Dit hoofdstuk geeft een introductie tot testen in het algemeen en tot gestructureerd testen in het bijzonder. Voor het kunnen begrijpen hiervan is geen specifieke TMap (voor)kennis vereist. Achtereenvolgens wordt een toelichting gegeven op: wat onder testen wordt verstaan, waarom testen nodig is (en wat het oplevert), wat de plaats van het testen is en wat gestructureerd testen is.

In hoofdstuk 3 “TMap in essenties” wordt de TMap invulling van een gestructureerde testaanpak beschreven.

2.1 Wat is testen?

Hoewel er vele definities over het begrip testen bestaan bevatten ze op één of andere wijze toch altijd vergelijkbare aspecten. In ieder van de definities staat een vergelijking van het object onder test met een norm (verwachting, correcte werking, eis) centraal. Daarbij is het van belang om te weten wat je gaat testen (testobject), waarmee je gaat vergelijken (testbasis) en hoe je gaat testen (testmethoden en -technieken).

De Internationale Standaardisatie Organisatie (ISO) en de International Electrotechnical Commission (IEC) hanteren de volgende definitie [ISO/IEC, 1991]:

Definitie

Testen bestaat uit activiteiten die uitgevoerd worden om één of meer kenmerken van een product, proces of dienst vast te stellen volgens een gespecificeerde procedure.

Testen[1] geeft inzicht in het verschil tussen de actuele en de vereiste status van een object. Waar kwaliteit grofweg te definiëren is als ‘voldoen aan de eisen en verwachtingen’, levert testen dus informatie op over de kwaliteit. Het geeft inzicht in bijvoorbeeld de risico’s die gelopen worden bij aanvaarding van mindere kwaliteit. Dat is dan ook de hoofddoelstelling van testen. Testen behoort tot de detectieve middelen van een kwaliteitssysteem. Het is familie van reviewing, simulatie, inspectie, auditing, keuring, desk-checking, walkthrough, enzovoort. De diverse detectie-instrumenten worden verdeeld in de groepen toetsen en testen:

- Toetsen: het beoordelen van tussenproducten.
- Testen: het beoordelen van de eindproducten.

Hard gesteld is bij het testen het vinden van fouten het hoofddoel: testen wil het gebrek aan kwaliteit aantonen, en dat openbaart zich in fouten. Formeel: het verschil tussen het product en de vooraf gestelde eisen vaststellen. Positief geformuleerd: vertrouwen schenken in het product.

Een goede of minder goede kwaliteit van producten heeft een relatie met de risico’s die een organisatie loopt bij het in gebruik nemen van deze producten. Daarom hanteren we binnen dit boek onderstaande definitie van testen volgens TMap:

Definitie

Testen is een proces dat inzicht geeft in- en adviseert over de kwaliteit en de daaraan gerelateerde risico’s.

Adviseren over de kwaliteit van wat? Voordat hier een antwoord op kan worden gegeven wordt eerst het begrip kwaliteit nader toegelicht. Want wat is eigenlijk kwaliteit?

Definitie

Kwaliteit is het geheel van eigenschappen en kenmerken van een product of dienst dat van belang is voor het voldoen aan vastgestelde of vanzelfsprekende behoeften [ISO, 1994].

Bij het streven om ‘vanzelfsprekende behoeften’ om te zetten in ‘vastgestelde behoeften’ stuit men al snel op het probleem dat het bespreekbaar maken van de kwaliteit van een informatiesysteem niet voor de hand ligt. Het ontbreekt aan een taal waarin men over kwaliteit kan praten. Maar sinds 1977, toen McCall [McCall, 1977] met het voorstel kwam om het concept kwaliteit onder te verdelen in een aantal verschillende kenmerken, de zogenaamde kwaliteitsattributen, is er op dit gebied veel progressie geboekt.

Definitie

Een kwaliteitsattribuut beschrijft een kenmerk van een informatiesysteem.

Een bekende set van kwaliteitsattributen is afkomstig van ISO en IEC [ISO 9126-1, 1999]. Daarnaast komt het vaak voor dat organisaties een eigen variatie op bovenstaande set maken of een eigen set samenstellen. Voor TMap is een specifiek voor testen geschikte set aan kwaliteitsattributen samengesteld. Deze wordt in hoofdstuk 10 “Kwaliteitsattributen en testvormen” opgesomd en toegelicht. Binnen het kader van dit boek wordt deze set gebruikt.

Wat is dan nu het antwoord op de vraag: “Adviseren over de kwaliteit van wat?”

Aangezien bij kwaliteit meestal aan het functioneel juist werken van de software wordt gedacht, kan samenvattend worden gezegd dat testen door velen wordt gezien als: vaststellen dat de software functioneel correct werkt. Hoewel dit in bepaalde situaties een goed antwoord kan zijn, dient men zich wel te realiseren dat testen meer is dan dat. Behalve de software zijn er meer testobjecten te benoemen waarvan de kwaliteit vastgesteld kan worden. Datgene waarover door testen een kwaliteitsadvies wordt gegeven, wordt testobject genoemd.

Definitie

Een testobject is het te testen (deel van het) informatiesysteem.

Het testobject kan bestaan uit hardware, systeemsoftware, applicatiesoftware, organisatie, procedures, documentatie of implementatie. En bij het adviseren over de kwaliteit hiervan kan het naast functionaliteit ook gaan over kwaliteitsattributen als bijvoorbeeld beveiliging, gebruikersvriendelijkheid, performance, onderhoudbaarheid, portabiliteit en testbaarheid.

Valkuilen

In de praktijk is het lang niet voor iedereen duidelijk wat testen is en wat er getest kan (moet) worden. Hier volgen enkele voorbeelden van wat testen *niet* is:

- Testen is niet het vrijgeven of accepteren. Het testen levert advies over de kwaliteit. De beslissing over vrijgave is aan anderen (stakeholders), veelal de opdrachtgever voor de test.
- Testen is niet een fase ná ontwikkeling. Het behelst een serie activiteiten die parallel moeten worden uitgevoerd aan ontwikkeling.
- Testen is iets anders dan het implementeren van een informatiesysteem.
  Testresultaten dwarsbomen de implementatieplannen nogal eens. Het is zaak deze toch vaak sterk gerelateerde activiteiten organisatorisch goed te beleggen.
- Testen is in eerste instantie niet bedoeld om vast te stellen of de juiste functionaliteit is gebouwd, maar om een belangrijke rol te spelen bij het vaststellen of de gewenste functionaliteit is gebouwd. Hoewel natuurlijk niet met oogkleppen op getest moet worden is de beoordeling of de goede oplossing is gespecificeerd een activiteit van een andere orde.
- Testen is niet goedkoop. Daarentegen zal een goede, tijdig uitgevoerde test het ontwikkelproces positief beïnvloeden en kan een kwalitatief beter systeem opgeleverd worden, waardoor tijdens productie minder storingen zullen optreden. Boehm heeft al lang geleden aangetoond dat het herstellen van fouten meer inspanning, tijd en geld kost naarmate het moment van detectie verder af ligt van het moment van ontstaan [Boehm, 1981]. Zie hiervoor ook de alinea “wat levert testen op?” in de volgende paragraaf.
- Testen is niet het opleiden voor gebruik en beheer. Omdat een testproces zich in het algemeen uitstekend voor dit doel leent, wordt dit aspect vaak te gemakkelijk als secundaire opdracht meegegeven. Goede afspraken moeten voorkomen dat zowel de test als de opleiding van onvoldoende kwaliteit zullen zijn. Er moet exclusief budget en tijd voor het opleiden beschikbaar worden gesteld en er moeten goede afspraken worden gemaakt over prioriteiten, want er zullen

mogelijk op bepaalde momenten keuzes moeten worden gemaakt.

Het is de taak van onder andere de testmanager om te voorkomen dat er in één van deze valkuilen wordt getrapt en om de opdrachtgever duidelijk te maken wat testen dan wel is.

2.2 Waarom testen?

In hoofdstuk 1 “Inleiding” is onder andere toegelicht dat IT sinds eind jaren negentig steeds belangrijker is geworden voor organisaties. Hierbij worden echter veel organisaties geplaagd door, zowel qua budget als tijd, uit de hand gelopen projecten en door softwarefouten tijdens de exploitatie van de ontwikkelde informatiesystemen. Daaruit blijkt dat organisaties systemen (moeten) accepteren zonder daadwerkelijk inzicht in de kwaliteit ervan. Daarnaast ontbreekt het bij de vrijgave in veel gevallen aan goede managementinformatie. Hierdoor worden vaak grote risico’s voor de bedrijfsvoering gelopen; hoge herstelkosten, productiviteitsverlies, schade aan het imago, schadeclaims en verlies van de concurrentiestrijd door te late beschikbaarheid van het nieuwe product (omzetverlies) kunnen het gevolg zijn.

Voordat een informatiesysteem in productie gaat, zal de organisatie zich expliciet af moeten vragen of aan alle voorwaarden is voldaan. Zijn alle delen en aspecten van het informatiesysteem met voldoende diepgang geraakt? Is behalve op de functionaliteit ook op bruikbaarheids-, prestatie- en bijvoorbeeld beveiligingsaspecten gecontroleerd? Of zoals ISO formuleert: Is vastgesteld of het product de eigenschappen en kenmerken bezit om te voldoen aan de vastgestelde of, en dat is nog lastiger, vanzelfsprekende behoeften?

Zijn alle fouten hersteld en zijn bij dat herstel geen nieuwe fouten geïntroduceerd? Ondersteunt het systeem de bedrijfsvoering? Biedt het systeem inderdaad de betrouwbare oplossing voor het informatieprobleem waarvoor het is ontworpen?

In feite luidt de vraag: Welke risico’s worden gelopen en welke maatregelen zijn genomen voor het verminderen van die risico’s. Om niet pas tijdens de exploitatiefase antwoord op dit soort cruciale vragen te krijgen is een goed en betrouwbaar testproces vereist. Dat vraagt om een gestructureerde testaanpak, organisatie en infrastructuur, waarmee op gecontroleerde wijze continu inzicht kan worden gegeven in de status van het systeem en de risico’s.

Wat levert testen op?

Hoewel een gestructureerde testaanpak van groot belang wordt geacht, volgt op de vraag “Wat vindt u van het testproces?” steevast het antwoord “duur!”. Dit antwoord kan zelden op waarde worden geschat, omdat het vaak gevoelsmatig wordt gegeven, waarbij cijfermatige onderbouwing ontbreekt. Testen is duur. Ja, dit is waar als hierbij alleen de testkosten, maar niet de testbaten in ogenschouw worden genomen.

Testkosten zijn onder andere gebaseerd op:

- de kosten van testinfrastructuur;
- de bestede uren van de testers en hun tarieven.

Testbaten zijn [Black, 2002]:

- Voorkomen van (hoge) herstelkosten en gevolgschade in de productiesituatie, doordat fouten tijdens het testen worden gevonden en binnen het systeemontwikkelproces worden opgelost. Voorbeelden van gevolgschade zijn: omzetverlies, imagobeschadiging, schadeclaims en productiviteitsverlies.
- Voorkomen van schade in productie, doordat fouten die tijdens het testen worden gevonden, niet worden opgelost, maar als ‘known errors’ bekend zijn.
- Vertrouwen hebben/krijgen in het product.
- Een goede projectbeheersing mogelijk maken door het leveren van (voortgangs- en kwaliteits)informatie.

Als er een manier is om de testbaten in geld uit te drukken, dan zou het antwoord op de vraag “wat levert testen op?” vanuit een testeconomisch perspectief kunnen zijn:

Testopbrengst = Testbaten - Testkosten

Hoewel dit een simpel rekensommetje lijkt, is het in de praktijk heel lastig om in te schatten wat fouten die tijdens testen worden gevonden voor schade in productie zouden hebben veroorzaakt als ze pas daar op zouden treden. Immers, hoe

vertaal je bijvoorbeeld een potentiële imagoschade naar geld? In de literatuur zijn echter aanpakken te vinden die een poging doen dit rekensommetje toch te kunnen maken (bijvoorbeeld: [Aalst, 1999]).

Feit blijft dat moeilijk is vast te stellen wat het vertrouwen hebben in de kwaliteit van een product of het krijgen van (voortgangs)informatie nu echt oplevert. Ondanks dat zijn er binnen de testwereld steeds meer tips en trucs te vinden waarmee het mogelijk is één of meer van onderstaande bevindingen te kunnen signaleren en om daarop te kunnen sturen:

- er wordt teveel getest waardoor de kosten om een fout tijdens test te vinden niet meer opwegen tegen de schade die deze fout in productie had kunnen veroorzaken als deze daar was opgetreden;
- er wordt te weinig getest waardoor meer incidenten in productie plaatsvinden en de herstelkosten hiervan in verhouding hoger zijn dan dat de testkosten waren geweest om deze fout tijdens test te vinden;
- er wordt ineffectief getest waardoor de testinvestering niet wordt terugverdiend.

2.3 Plaats van het testen

Deze paragraaf geeft helderheid over zowel de betekenis als plaats van bepaalde testbegrippen in hun omgeving. Verdeeld over de volgende onderwerpen worden de daarmee samenhangende begrippen toegelicht:

- Testen en kwaliteitszorg
- Testen, hoe en door wie
- Test- en systeemontwikkelproces
- Testsoorten en -verantwoordelijkheden
- Testvormen

2.3.1 Testen en kwaliteitszorg

Kwaliteit was, is en blijft vooralsnog een uitdaging binnen de automatiseringsbranche. (zie ook voorbeelden in paragraaf 1.2 “TMap evolueert mee”)

Testen is daarvoor niet dé oplossing. Immers, kwaliteit moet er in worden gebouwd en kan er niet in worden getest! Testen is hét instrument dat inzicht kan geven in de kwaliteit van informatiesystemen en daardoor leveren testresultaten, mits goed geïnterpreteerd, een bijdrage in het streven de kwaliteit van informatiesystemen te verbeteren. Testen dient te worden ingebed in een systeem van maatregelen om tot kwaliteit te komen. Met andere woorden: testen dient te worden ingebed in de kwaliteitszorg van de organisatie.

Uit de definitie van kwaliteit volgens ISO spreekt heel sterk het ongrijpbare ervan. Wat voor de één vanzelfsprekend is, is dat voor de ander absoluut niet. Vanzelfsprekendheid is bij uitstek een subjectief begrip. Een belangrijk aspect van kwaliteitszorg is daarom het minimaliseren van vanzelfsprekende behoeften, door deze om te zetten in vastgestelde behoeften (eisen) en zichtbaar te maken in welke mate aan de vastgestelde behoeften wordt voldaan.

Het structureel verbeteren van kwaliteit moet topdown gebeuren. Daartoe moeten maatregelen getroffen worden om die eisen vast te stellen en het ontwikkelproces beheersbaar te maken.

Definitie

Kwaliteitsborging (quality assurance) omvat het geheel van alle geplande en systematische acties nodig om in voldoende mate het vertrouwen te geven dat een product of dienst voldoet aan de gestelde kwaliteitseisen [ISO, 1994].

Deze maatregelen moeten er toe leiden dat:

- er meetpunten en -grootheden zijn die een indicatie geven van de kwaliteit van de processen (normering);
- het voor de individuele medewerker duidelijk is aan welke eisen zijn werk moet voldoen en dat hij deze aan de hand van bovengenoemde normen kan toetsen;
- het voor een onafhankelijke partij mogelijk is de producten/diensten aan de hand van bovengenoemde normen te toetsen;
- het management bij gebleken gebreken in (deel-)producten of diensten kan traceren waardoor die gebreken zijn

ontstaan en hoe deze in de toekomst kunnen worden voorkomen.

Deze maatregelen zijn te onderscheiden naar preventieve, detectieve en correctieve maatregelen:

- Preventieve maatregelen hebben tot doel gebrek aan kwaliteit te voorkomen. Hierbij kan gedacht worden aan documentatienormen, methoden, technieken, opleidingen, enzovoort.
- Detectieve maatregelen hebben tot doel gebrek aan kwaliteit te ontdekken, door bijvoorbeeld toetsen (o.a.: inspecties, reviews, walkthroughs) en testen.
- Correctieve maatregelen hebben tot doel gebrek aan kwaliteit op te heffen, zoals het herstellen van fouten die door middel van testen zijn blootgelegd.

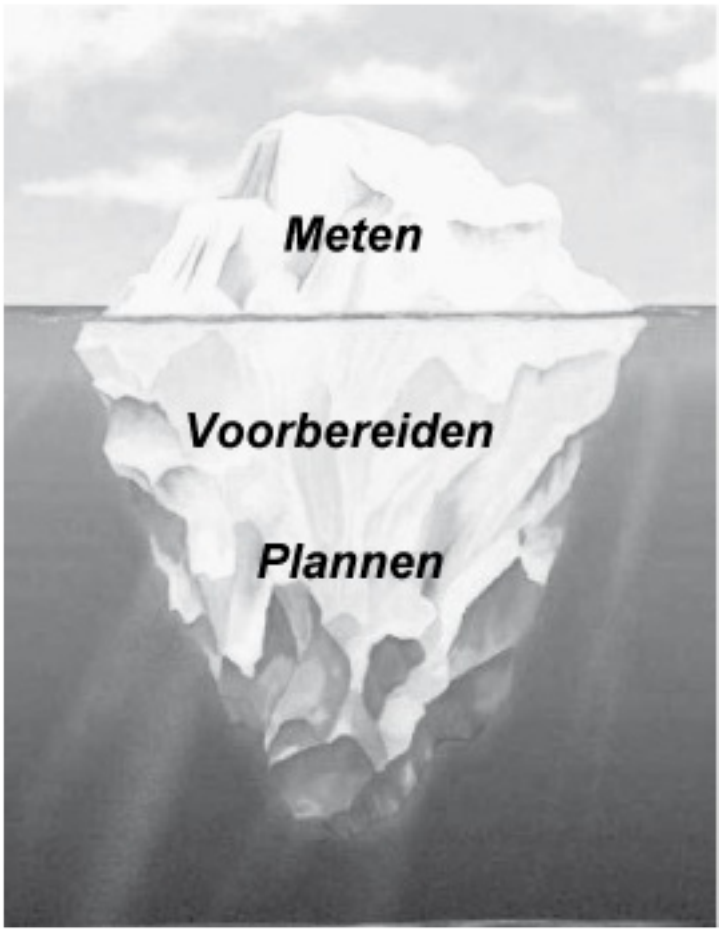
Het is van wezenlijk belang dat er samenhang bestaat tussen de verschillende maatregelen. Testen is niet een alleenstaande activiteit. Testen is slechts een klein radertje in het bouwwerk van de zorg om kwaliteit. Het is slechts één van de vormen van kwaliteitsdetectie die gebruikt kunnen worden. Kwaliteitsdetectie is op haar beurt slechts één van de activiteiten om kwaliteit te borgen. En kwaliteitsborging is tenslotte slechts één van de dimensies van kwaliteitszorg.

2.3.2 Testen, hoe en door wie?

Testen krijgt vaak weinig aandacht tot het moment dat er getest gaat worden. Dan komen er opeens heel veel geïnteresseerden bij de testmanager informeren hoe het er voor staat. Deze paragraaf laat zien dat testen echter meer is dan alleen het uitvoeren van tests. Vervolgens wordt toegelicht op welke manieren en door wie er kan worden getest.

Testen is meer dan uitvoeren

Testen is niet alleen het uitvoeren van tests (meten), maar vooral ook plannen en voorbereiden. Het testen is te vergelijken met een ijsberg. Slechts een klein gedeelte van een ijsberg, het topje, bevindt zich boven de zeespiegel. Het grootste gedeelte van de ijsberg bevindt zich echter onder de zeespiegel en wordt niet direct herkend (zie figuur 2.1 “De ijsberg”).



Figuur 2.1: De ijsberg.

In deze vergelijking wordt het daadwerkelijke uitvoeren van de tests gezien als het zichtbare deel van testen, terwijl dit gemiddeld maar 40% van de testactiviteiten behelst. De overige activiteiten, plannen en voorbereiden, vragen respectievelijk gemiddeld 20% en 40% van de testinspanning. Dit deel wordt meestal niet als zodanig door de organisatie onderkend, terwijl hiermee nu juist de meeste (tijd-)winst uit het testen te halen is. Én niet onbelangrijk, door deze activiteiten zoveel mogelijk voor het daadwerkelijk uitvoeren van de tests uit te voeren, bevindt testen zich zo kort als mogelijk op het kritieke pad van het systeemontwikkeltraject. Het is zelfs zo dat door technische ontwikkelingen (testautomatisering) een tendens waarneembaar is, waarbij het percentage uitvoeren ten opzichte van voorbereiden en plannen verder daalt.

Manieren van testen

Bij testen, in dit geval bij het uitvoeren van tests, zijn er diverse manieren waarop kan worden getest. Wordt er bijvoorbeeld getest door het laten runnen van software, of juist niet? En wordt er een kenmerk van het systeem getest met speciaal hiervoor ontworpen testgevallen of juist niet? Een aantal manieren van testen zijn:

- Dynamisch expliciet testen
- Dynamisch impliciet testen
- Statisch testen

Dynamisch expliciet testen

Bij dynamisch expliciet testen zijn de testgevallen expliciet ontworpen om informatie over het betreffende kenmerk (kwaliteitsattribuut) te verkrijgen. Door executie van het testobject dan wel het runnen van software wordt het actuele resultaat vergeleken met het verwachte resultaat om zo te bepalen of het systeem zich als vereist gedraagt. Dit is de meest gangbare manier van testen.

Dynamisch impliciet testen

Tijdens het dynamisch testen kan ook informatie over andere kenmerken (kwaliteitsattributen) worden verzameld, waar niet expliciet testgevallen voor zijn ontworpen. Dit wordt dynamisch impliciet testen genoemd. Zo kunnen over bijvoorbeeld de gebruiksvriendelijkheid of performance van een systeem uitspraken worden gedaan op basis van tijdens het testen opgedane ervaringen zonder dat daarvoor gerichte testgevallen aanwezig waren. Dit kan gepland gebeuren in het geval er vóóraf afgesproken was om hierover een uitspraak te moeten doen. Maar dit kan ook ongepland gebeuren. Bijvoorbeeld in het geval dat tijdens het testen regelmatig systeemstoringen optreden. Dan kan er een uitspraak worden gedaan over de bedrijfszekerheid.

Statisch testen

Bij statisch testen worden de eindproducten beoordeeld zonder dat er sprake is van het runnen van software. Deze test bestaat meestal uit het inspecteren van documentatie zoals beveiligingsprocedures, opleidingen, handleidingen, enzovoort en wordt vaak met checklists ondersteund.

Wie test er?

In feite kan er door iedereen worden getest. Wie er test wordt mede bepaald door de rol of verantwoordelijkheid die iemand op een bepaald moment heeft. Vaak zijn dit vertegenwoordigers uit ontwikkel-, gebruikers- en/of beheerafdelingen. Hiernaast wordt er getest door beroepstesters. Deze in het testvak getrainde professionals hebben vaak een andere kijk op testen. Waar bijvoorbeeld een ontwikkelaar aan wil tonen dat de software goed werkt (“ik zal toch zeker wel kunnen programmeren?”), zal de testprofessional op zoek gaan naar fouten in de software. Verder is een testprofessional fulltime bezig met testen terwijl de eerder genoemde afdelingsvertegenwoordigers het testen in veel gevallen erbij doen. In de praktijk blijkt een mix van goed opgeleide testprofessionals en vertegenwoordigers uit de diverse afdelingen tot een goede wisselwerking te leiden, waarbij de één sterk is in kennis en kunde over testen en de ander veel materie- of systeemdeskundigheid met zich meebrengt.

2.3.3 Test- en systeemontwikkelproces

Het test- en systeemontwikkelproces zijn nauw met elkaar verweven. De één levert de producten, die door de ander getoetst en getest worden. Een veel gehanteerde manier om de relatie tussen deze processen te visualiseren is het zogenaamde V-model. Een wijd verbreid misverstand hierbij is dat het V-model een watervalmethode zou representeren. Zo is het model echter niet bedoeld. Het model is ook heel goed bruikbaar bij een iteratieve en incrementele systeemontwikkelmethode. Bij een dergelijke methode kan dan bijvoorbeeld voor elk increment een V-model worden getekend. Er zijn vele situaties te bedenken die invloed hebben op de vorm en de specifieke onderdelen van het V-model. Enkele situaties worden gegeven in onderstaand kader “invloeden op het V-model”.

Met behulp van het V-model wordt in deze en de volgende paragraaf de samenhang tussen testbasis, toetsen en testen (testsoorten) toegelicht.

Uitgediept

Invloeden op het V-model

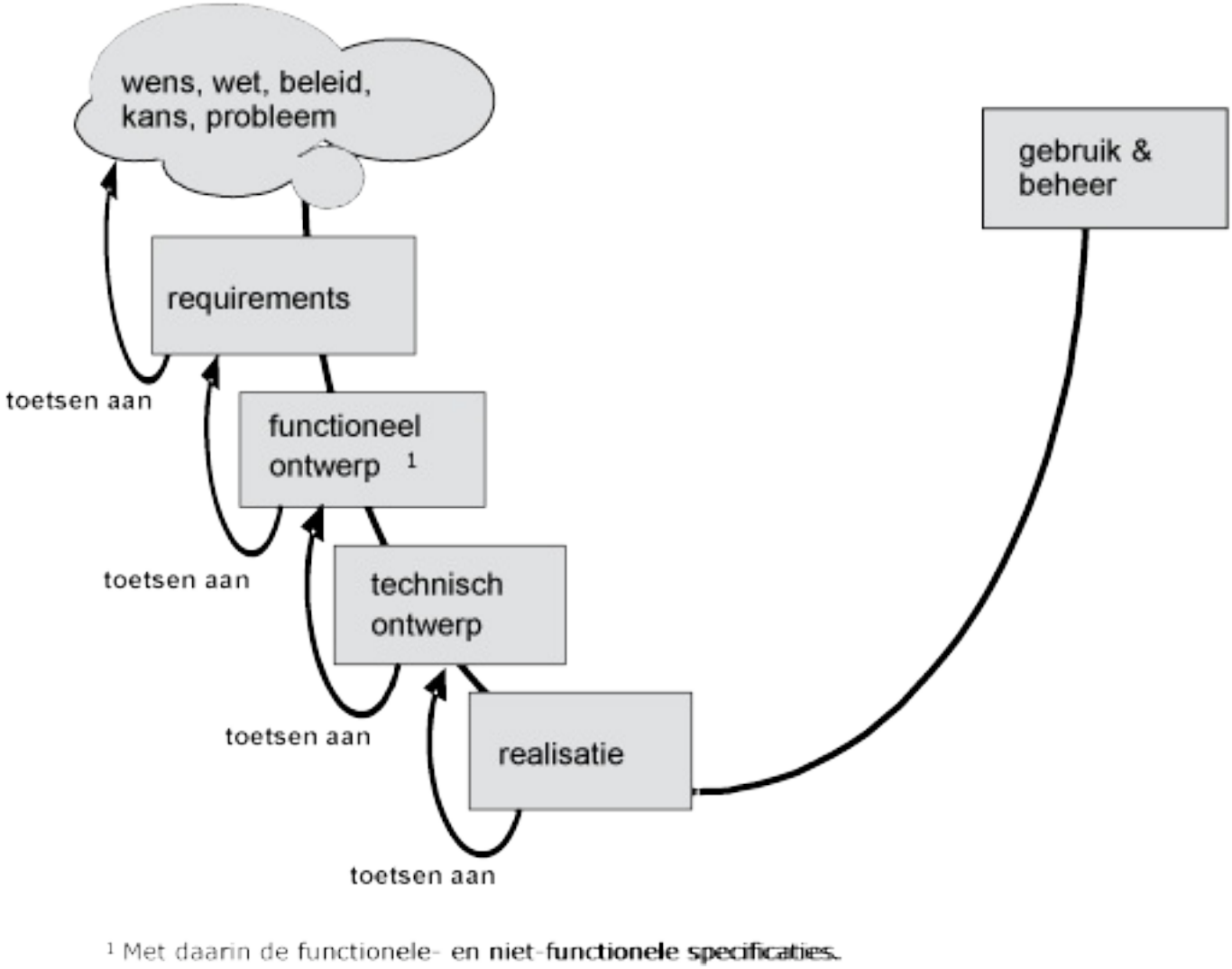
Vorm en specifieke onderdelen van een V-model kunnen variëren door bijvoorbeeld:

- De plaats van het testen binnen de systeemontwikkelaanpak.
  - Bij gebruik van een waterval ontwikkelmethode met onder andere de kenmerken: het in één keer bouwen van het systeem, gefaseerd met duidelijke overdrachtsmomenten, vaak een langdurig cyclisch proces (o.a. SDM).
  - Bij het gebruik van een incrementele en iteratieve ontwikkelmethode met volgende mogelijke kenmerken: het in gedeelten bouwen van het systeem, gefaseerd met duidelijke overdrachtsmomenten; kort cyclisch proces (o.a. DSDM en RUP).
  - Bij het gebruik van een agile ontwikkelmethode gekenmerkt door de vier principes; individuen en interactie boven processen en tools, werkende software boven uitgebreide systeemdocumentatie, gebruikersinbreng boven contractonderhandeling, reageren op veranderingen boven het volgen van een plan (o.a. extreme programming en SCRUM).
- De plaats van het testen binnen de levenscyclus van het informatiesysteem.
  - Betreft het hier de nieuwbouw van- of het onderhoud op een systeem?
  - Betreft het hier de conversie of migratie van een systeem?
- Een zelfontwikkeld systeem, of een aangeschaft pakket, of gekochte componenten, of gedistribueerde systemen.
- De situatie dat (delen van) systeemontwikkeling en/of (delen van het) testen zijn uitbesteed (o.a. outsourcing en off-/near shoring).

Linkerkant van het V-model

In figuur 2.2 “V-model (de linkerkant)” staan aan de linkerkant de fasen waarin het systeem wordt gebouwd of verbouwd, van wens, wet, beleid, kans en/of probleem naar de gerealiseerde oplossing.

In dit geval staan aan de linkerzijde de begrippen requirements, functioneel-, technisch ontwerp en realisatie. Hoewel de exacte naamgeving van deze begrippen afhankelijk is van de gekozen ontwikkelmethode, is deze niet nodig om op globaal niveau de relatie tussen het systeemontwikkel- en testproces aan te kunnen geven.



Figuur 2.2: V-model (de linkerkant).

Toetsen

Tijdens het systeemontwikkelproces worden diverse tussenproducten ontwikkeld. Deze hebben afhankelijk van de gekozen methode een bepaalde vorm, inhoud en een relatie met elkaar en kunnen daarop worden getoetst.

> **Definitie**
>
> Toetsen is het beoordelen van de tussenproducten in het systeemontwikkelproces.

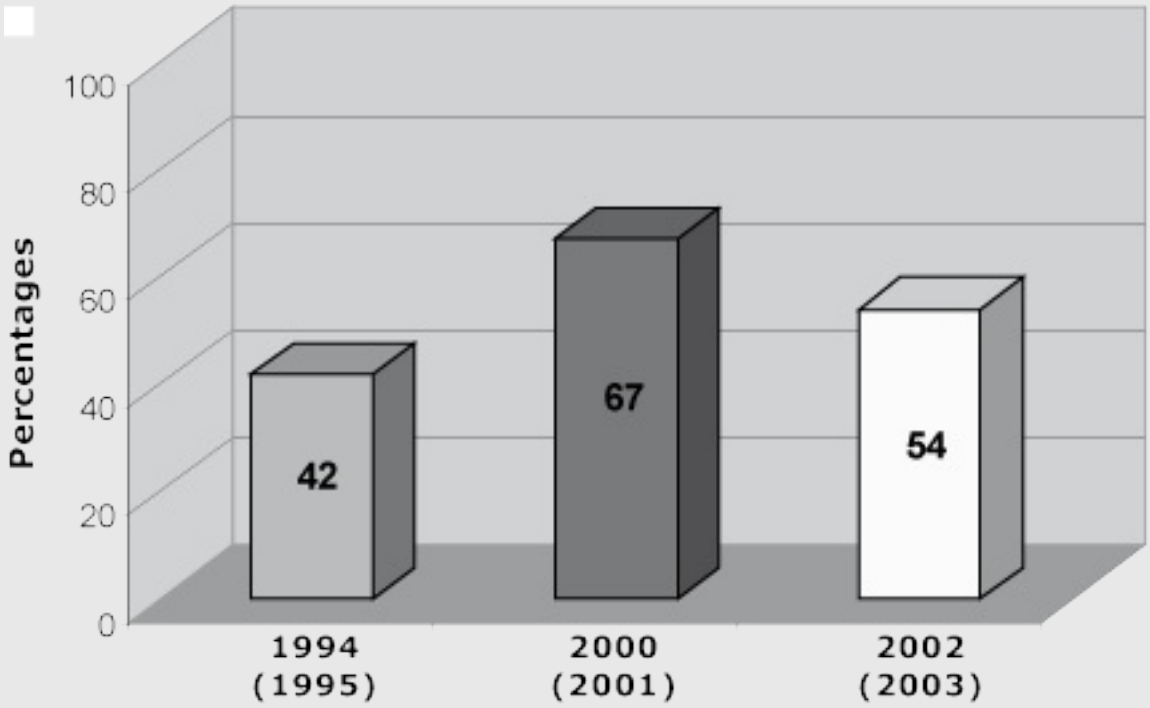
In het V-model is aan de linkerkant te zien welke tussenproducten (aan elkaar) getoetst kunnen worden. Bij het toetsen kan het resultaat worden vergeleken met:

- Het voorgaande tussenproduct
  Is bijvoorbeeld het functioneel ontwerp consistent met het technisch ontwerp of zijn de specificaties testbaar?
- De eisen vanuit het vervolgtraject
  Kan bijvoorbeeld de ontwikkelaar het gegeven ontwerp eenduidig realiseren?
- Andere tussenproducten op hetzelfde niveau
  Is bijvoorbeeld het functioneel ontwerp intern consistent en met daaraan gerelateerde functioneel ontwerpen?
- De afgesproken productstandaard
  Zijn er bijvoorbeeld use cases aanwezig?
- De verwachtingen van de opdrachtgever (zie kader “gerealiseerde requirements”)
  Is het tussenproduct nog consistent met de verwachtingen van de acceptanten?

Hierbij zijn voor het toetsen diverse technieken beschikbaar: reviews, inspecties en walkthroughs (zie ook hoofdstuk 15 “Toetstechnieken”).

Uitgediept

Gerealiseerde requirements



Figuur 2.3: Gerealiseerde requirements.

Hoe zit dat nu precies met het traject van wens, wet, enzovoort naar product? Worden bijvoorbeeld alle requirements gerealiseerd of gaat er onderweg nog wel eens iets verloren? Een door de Standish Group uitgevoerd onderzoek laat helaas een weinig florissant beeld zien. De bevindingen van het onderzoek (figuur 2.3 “Gerealiseerde requirements”), waarin het percentage gerealiseerde requirements werd bepaald, laat zien dat van de oorspronkelijk gedefinieerde requirements slechts 42% tot 67% daadwerkelijk door het project wordt gerealiseerd [The Standish Group, 2003].

Naast normale toetsresultaten (het vinden van fouten) kan een goed ingericht en uitgevoerd toetsproces ook een bijdrage leveren aan een hoger realisatiepercentage van de oorspronkelijk gedefinieerde requirements.

2.3.4 Testsoorten en -verantwoordelijkheden

In een systeemontwikkeltraject kan een scheiding worden aangebracht in verantwoordelijkheden tussen opdrachtgever, gebruiker, beheerder, systeembeheer enerzijds en systeemontwikkelaar, leverancier anderzijds. In de context van het testen wordt de eerste groep samengevat als de accepterende (vragende) partij en de tweede groep als leverende partij. Andere begrippen die in dit verband ook wel worden genoemd zijn demand- en supply organisatie. Op globaal niveau zijn er bij testen twee mogelijke doelen die nagestreefd kunnen worden:

- De leverende partij toont aan dat wat geleverd moet worden ook werkelijk wordt geleverd.
- De accepterende partij stelt vast of wat gevraagd is ook werkelijk wordt verkregen en of ze met het product kunnen doen wat ze willen/moeten doen.

Rechterkant van het V-model

In figuur 2.4 “V-model (de rechterkant)” geeft een horizontale stippellijn deze (formele) scheiding aan. In de praktijk is deze scheiding minder hard en zullen systeemontwikkelaars worden ingeschakeld bij de informatieanalyse en bij het opstellen van de specificaties en leveren zij ondersteuning bij de acceptatietest. Ook zal expertise van gebruikers en beheerders worden gebruikt bij bouwactiviteiten. Het is van belang de verschillende verantwoordelijkheden goed vast te stellen. Dit geldt zeker ook voor het testen. Wie fungeert als opdrachtgever voor een test, wie accepteert, wie wil advies over de kwaliteit en wie test wat wanneer?

Aan de rechterkant van het V-model vindt het testen plaats. Hierbij wordt binnen de leverende partij vaak onderscheid gemaakt tussen testen door de ontwikkelaar en testen door project/leverancier:

- Testen door ontwikkelaar
  Bijvoorbeeld door een programmeur, technisch ontwerper of engineer.
- Testen door project/leverancier
  Bijvoorbeeld door een project of leverancier van software of pakketleverancier of onderhoudsorganisatie.

Dit onderscheid in (test)verantwoordelijkheden wordt in de praktijk vertaald in het groeperen van testactiviteiten in testsoorten.

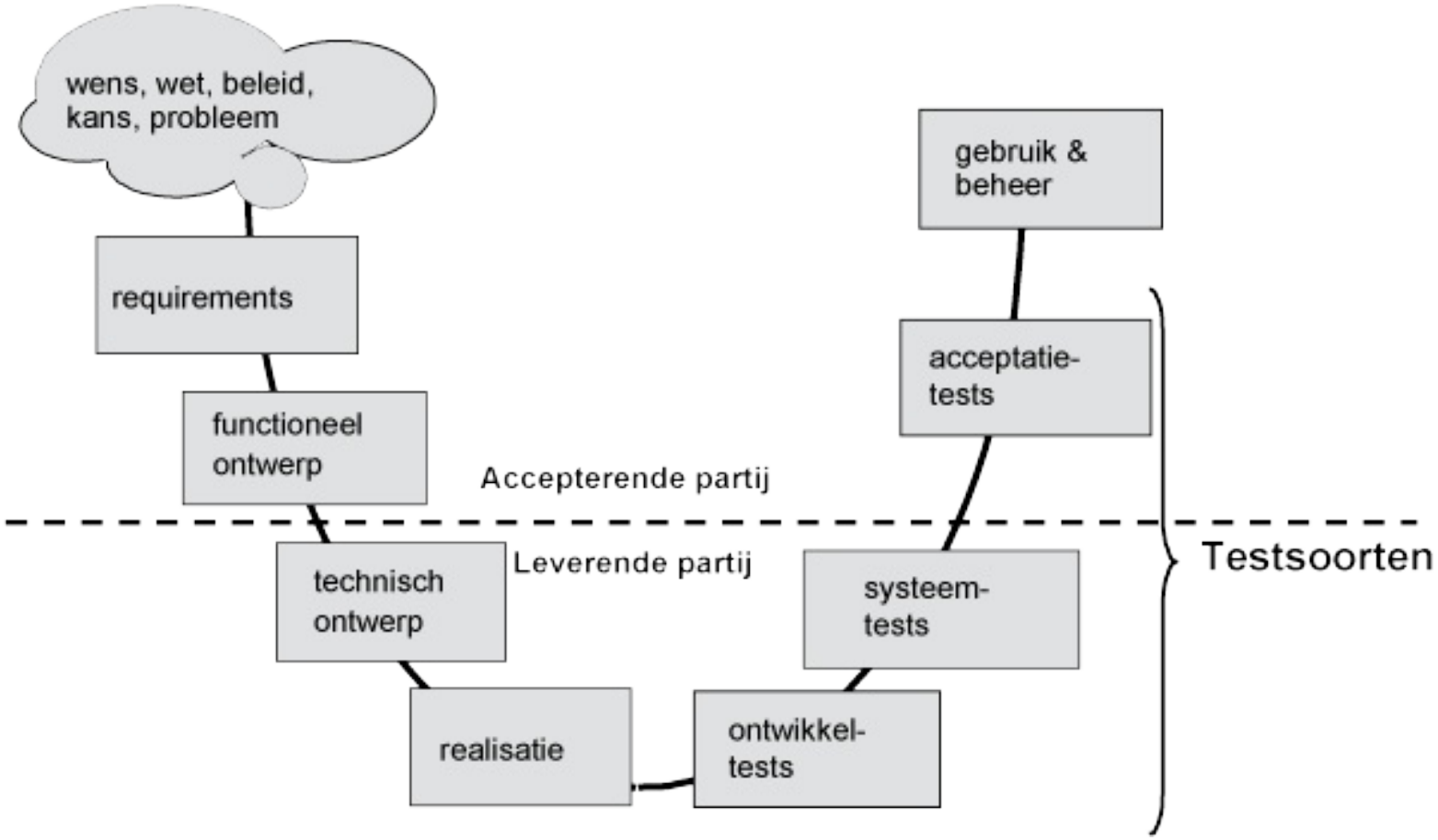
Definitie

Een testsoort is een groep van testactiviteiten die gezamenlijk worden uitgevoerd en aangestuurd.

Tegenover iedere fase van het bouwen staan een of meer testsoorten. Een misverstand hierbij is dat de testsoortrechthoekjes in het V-model als fasen van het systeemontwikkelproces worden beschouwd. Deze vertegenwoordigen echter de testuitvoering (de meetfase) van een testsoort.

Figuur 2.4 “V-model (de rechterkant)” vermeldt de ontwikkel- en systeemtests onder verantwoordelijkheid van de leverende partij en de acceptatietests onder verantwoordelijkheid van de accepterende partij:

- Ontwikkeltests
  Tests waarin de ontwikkelende partij aantoont dat het product voldoet aan met name de technische specificaties.
- Systeemtests
  Tests waarin de leverende partij aantoont dat het product voldoet aan met name de functionele- en niet-functionele specificaties en aan het technisch ontwerp.
- Acceptatietests
  Tests waarin de accepterende partij vaststelt dat het product voldoet aan met name de verwachtingen.



Figuur 2.4: V-model (de rechterkant).

Hoewel testsoort een veel gebruikt begrip is komt het in de praktijk vaak voor dat men moeite heeft met de invulling

daarvan. In de praktijk blijkt dat het niet mogelijk is om *de* testsoortenset te benoemen. Zelfs binnen één bedrijf lukt het vaak niet om één set te definiëren die in alle projecten gebruikt zou moeten worden.

In dit boek worden de genoemde drie testsoorten gebruikt. In voorkomende gevallen, om een bepaalde situatie beter te kunnen beschrijven, worden deze testsoorten nog verder uitgesplitst. In hoofdstuk 4 “Inleiding processen” wordt de door TMap gebruikte set van testsoorten nader benoemd.

Zoals eerder vermeld is er niet één standaard set van testsoorten te benoemen. Simpelweg omdat dat erg afhankelijk is van de organisatie, van het project en zelfs van de persoon. Maar uiteraard zijn er wel enkele overwegingen op te sommen om in een bepaalde situatie tot een daarop toegesneden testsoortenindeling te komen. Deze overwegingen kunt u lezen in hoofdstuk 5 “Mastertestplan, managen van het totale testproces”.

Testbasis en testsoorten

Een testsoort heeft als doel om aan te tonen in welke mate het product aan bepaalde verwachtingen, eisen, functionele specificaties of technische specificaties voldoet. Als referentie wordt hierbij vaak systeemdocumentatie gebruikt. In bepaalde situaties, meestal in het geval van een migratietraject, kan als referentie ook het huidige productiesysteem dienen. Als er weinig, geen of verouderde systeemdocumentatie is, kan ook de kennis van bijvoorbeeld eindgebruikers en producteigenaars als referentie worden gebruikt. Er zijn vele informatiebronnen die als referentie bij het testen kunnen worden gebruikt. De algemene term hiervoor is testbasis.

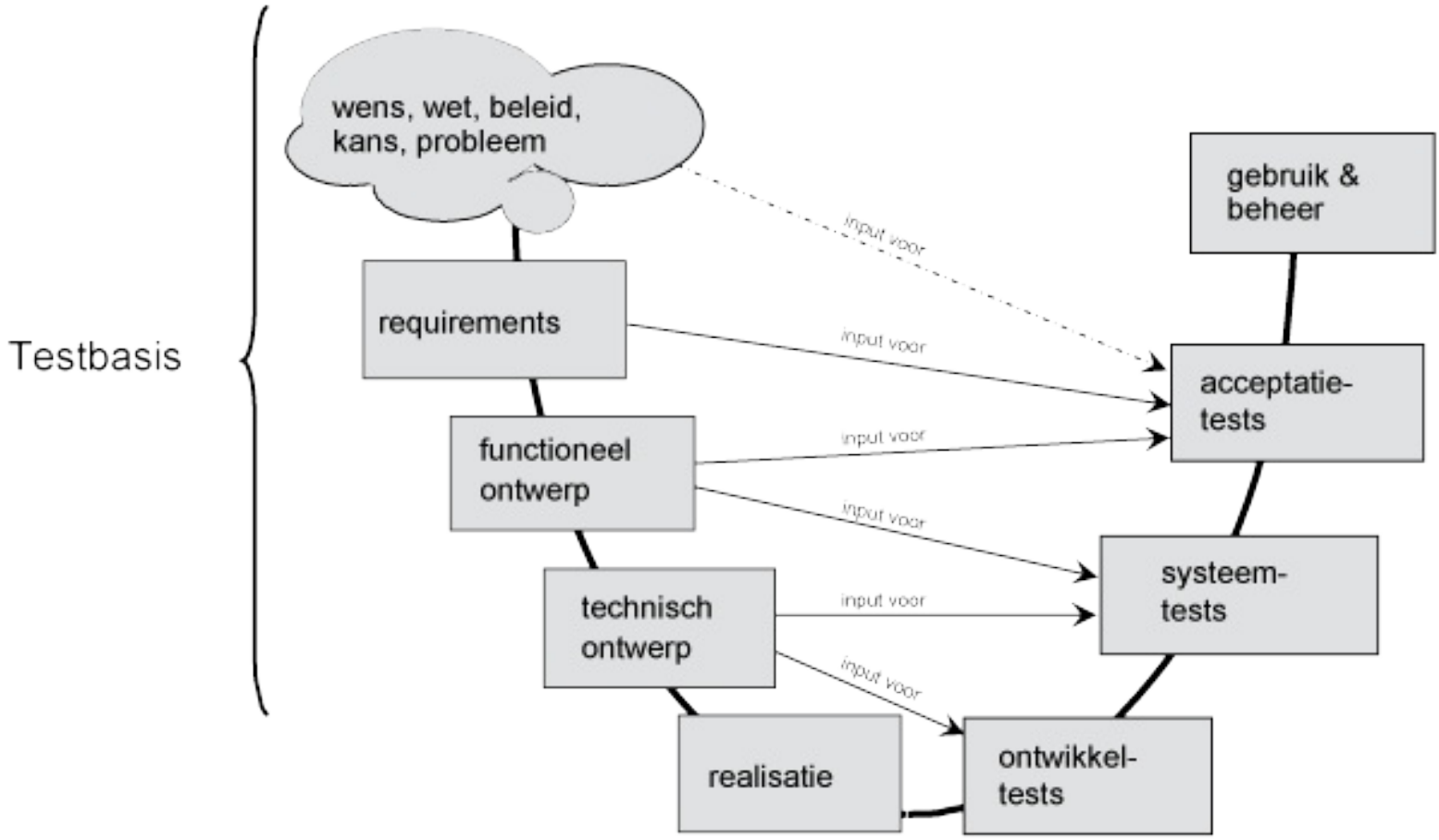
Definitie

De testbasis is de informatie die het gewenste systeemgedrag definieert.

Deze wordt als basis gebruikt voor het creëren van testgevallen. Indien een testbasis nog slechts via de formele wijzigingsprocedure kan worden gewijzigd, spreekt men over een gefixeerde testbasis.

Figuur 2.5 “V-model (testbasis)” is door middel van pijlen met de tekst “input voor” aangegeven welke informatiebronnen in welke testsoort als basis kunnen worden gebruikt om testgevallen van af te leiden. Uit het model blijkt ook dat het mogelijk is dat dezelfde testbasis in twee testsoorten wordt gebruikt. Vaak gebeurt dit als er twee verschillende partijen vanuit hun eigen verantwoordelijkheid een test moeten uitvoeren. In het getekende model wordt een functioneel ontwerp als testbasis gebruikt door de leverende partij om aan de accepterende partij bijvoorbeeld aan te kunnen tonen dat het systeem voldoet aan dit ontwerp. De accepterende partij echter gebruikt hetzelfde functioneel ontwerp als testbasis om daarmee te kunnen controleren of het door de leverancier opgeleverde systeem daadwerkelijk aan dit ontwerp voldoet.

Het zal duidelijk zijn dat in een dergelijke situatie de kans aanwezig is dat er dubbel wordt getest. Dat kan overigens een bewuste en volkomen valide keuze zijn, maar het kan net zo terecht zijn om bepaalde testsoorten samen te nemen. In hoofdstuk 5 “Mastertestplan, managen van het totale testproces” vindt u een aantal overwegingen om daar een keuze in te kunnen maken.



Figuur 2.5: V-model (testbasis).

2.3.5 Testvormen

Tijdens het testen van een systeem kan naar verschillende soorten kenmerken worden gekeken; de zogenaamde kwaliteitsattributen. Voorbeelden hiervan zijn functionaliteit, performance en continuïteit.

Op detailniveau kan het echter zijn dat een bepaald kwaliteitsattribuut een té algemeen kenmerk is, waardoor deze in de praktijk moeilijk hanteerbaar blijkt te zijn. Het kwaliteitsattribuut moet dan in een vorm worden gegoten waarmee de tester beter uit de voeten kan. In het voorbeeld van functionaliteit kan dan worden gedacht aan risico's op het gebied van interfaces, in- en uitvoer validaties, relatiecontroles, of uitsluitend de verwerking. Bij performance kan worden gedacht aan load en/of stress risico's. En in het voorbeeld van continuïteit kan worden gedacht aan het belang van backup-, restore- en/of uitwijkfaciliteiten. Deze vaak gebruikte vorm van kwaliteitsattributen wordt testvorm genoemd.

Definitie

Een testvorm is een groep testactiviteiten met het oogmerk het informatiesysteem op een aantal samenhangende (deelaspecten van) kwaliteitsattributen te controleren.

Op de website www.tmap.net vindt u een lijst van een aantal veel voorkomende testvormen. Deze opsomming is niet uitputtend en zal variëren van testproject tot testproject en van organisatie tot organisatie.

Een vreemde eend in de bijt in deze opsomming is de testvorm regressie. Immers een testvorm was nu net bedoeld om detailinformatie te geven over de specifieke risico's met betrekking tot de betreffende kwaliteitsattributen. Terwijl regressie daarentegen juist een vrij globale term is en eigenlijk zelf een specifiek risico benoemt.

Definitie

Regressie is het verschijnsel dat de kwaliteit van een systeem als geheel achteruit gaat als gevolg van individuele aanpassingen.

Definitie

Een regressietest is erop gericht om te controleren dat alle ongewijzigde onderdelen van een systeem nog correct functioneren na het doorvoeren van een wijziging.

Vaak is het vaststellen of er geen regressie is opgetreden een doel op zich.

Het is daarom beter om hier al aandacht aan te besteden als er op globaal niveau nagedacht wordt over het verdelen van de kwaliteitsattributen over testsoorten.

Bij de detailinvulling kan dan worden nagedacht over wat wordt bedoeld met ‘correct functioneren’ in bovenstaande definitie. Gaat het hierbij om bijvoorbeeld functionaliteit, performance of uitwijkfaciliteiten? In feite kunnen alle kwaliteitsattributen en testvormen onderdeel van een regressietest zijn.

Hoe testvormen binnen TMap worden gebruikt, wordt toegelicht in de hoofdstukken 6 “Acceptatie- en systeemtesten” en 10 “Kwaliteitsattributen en testvormen”.

2.4 Wat is gestructureerd testen?

In de praktijk blijkt dat in veel projecten nog steeds op ongestructureerde wijze wordt getest. Naast een aantal uitingen (nadelen) van ongestructureerd testen en (voordelen) van gestructureerd testen worden in deze paragraaf enkele kenmerken van een gestructureerde testaanpak gegeven.

Nadelen van ongestructureerd testen

Ongestructureerd testen kenmerkt zich door een ongeordende toestand, waarin het onmogelijk is om de testinspanning te kunnen voorspellen, tests herhaalbaar uit te kunnen voeren of resultaten te meten. Vaak wordt dit ad hoc testen genoemd. In een dergelijke aanpak worden geen kwaliteitscriteria gebruikt om bijvoorbeeld risico’s en testactiviteiten te kunnen bepalen en te prioriteren. Ook wordt er voor het maken van testgevallen geen gebruik gemaakt van een testontwerptechniek. Enkele bevindingen die uit de diverse onderzoeken naar (on)gestructureerd testen naar voren zijn gekomen zijn:

- Tijdsdruk door:
  - het ontbreken van een goede testplanning en begrotingsmethode;
  - het ontbreken van een aanpak waarin staat beschreven welke testactiviteiten in welke fase door wie moeten worden uitgevoerd;
  - het ontbreken van goede afspraken over termijnen en procedures voor oplevering en herstel van de applicaties.
- Geen inzicht en advies kunnen geven over de kwaliteit van het systeem door:
  - het ontbreken van een risicoanalyse;
  - het ontbreken van een teststrategie;
  - het niet gebruiken van testontwerptechnieken, waardoor zowel kwaliteit als kwantiteit van de testgevallen onvoldoende is.
- Inefficiënt en ineffectief door:
  - het ontbreken van afstemming tussen de diverse testpartijen, waardoor er mogelijk objecten dubbel worden getest of erger nog: helemaal niet worden getest;
  - het ontbreken van afspraken op het gebied van configuratie- en wijzigingenbeheer voor zowel test- als systeemontwikkelproducten;
  - het niet- of onjuist gebruiken van de vaak wel aanwezige testhulpmiddelen;
  - het ontbreken van prioriteiten zodat minder belangrijke delen vaak eerder worden getest dan meer risicovolle delen.

Voordelen gestructureerde testaanpak

Maar wat zijn dan de voordelen van gestructureerd testen? Een flauw, maar wel correct, antwoord is dat in een gestructureerde testaanpak de genoemde nadelen ontbreken. Of, positief geformuleerd, een gestructureerde testaanpak biedt de volgende voordelen:

- is inzetbaar in elke situatie, ongeacht wie de opdrachtgever is of welke systeemontwikkelaanpak wordt gebruikt;

- geeft inzicht in- en advies over eventuele risico's ten aanzien van de kwaliteit van het geteste systeem;
- vindt fouten in een vroeg stadium;
- voorkomt fouten;
- het testen bevindt zich zo kort mogelijk op het kritieke pad van de totale ontwikkeling, waardoor de totale doorlooptijd van de ontwikkeling korter wordt;
- de testproducten (bijvoorbeeld testgevallen) zijn herbruikbaar;
- het testproces is inzichtelijk en beheersbaar.

Kenmerken gestructureerde testaanpak

Hoe ziet een gestructureerde testaanpak er dan uit? Hiervoor zijn vele invullingen te bedenken. In hoofdstuk 3 "TMap in essenties" en daarop volgende hoofdstukken wordt de specifieke TMap invulling hiervan gegeven.

In het algemeen kan worden gezegd dat een gestructureerde testaanpak wordt gekenmerkt door:

- Het bieden van structuur, zodat duidelijk is wat, door wie, wanneer en in welke volgorde moet worden gedaan.
- Het omvatten van de volledige scope en de beschrijving van het complete scala aan relevante aspecten.
- Het bieden van concrete handvatten, zodat niet steeds opnieuw het wiel uitgevonden hoeft te worden.
- Het sturen van de testactiviteiten in het kader van tijd, geld en kwaliteit.

Noot

1 In de theorie wordt ook wel gesproken van verificatie en validatie. Verificatie houdt dan in het evalueren van (een deel van) een systeem om te bepalen of de producten van een ontwikkelingsfase voldoen aan de voorwaarden die gesteld zijn aan het begin van die fase. Onder validatie wordt verstaan het bepalen of de producten van de systeemontwikkeling voldoen aan de behoeften en eisen van de gebruikers. [IEEE, 1998]

3 TMap in essenties

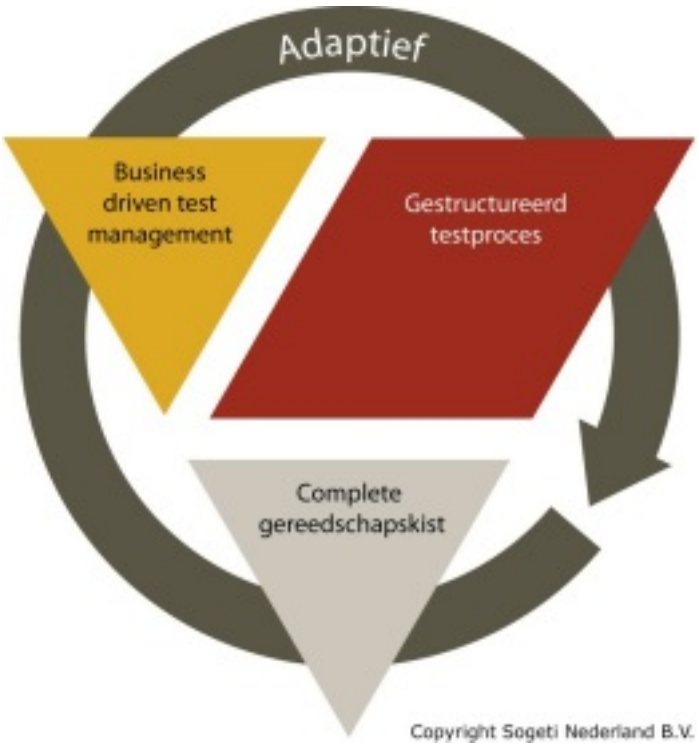
In dit hoofdstuk wordt de specifieke TMap invulling van een gestructureerde testaanpak gegeven. Deze invulling is in een viertal essenties samen te vatten.

De vier essenties van TMap

1. TMap is gebaseerd op een business driven testmanagement (BDTM) aanpak.
2. TMap beschrijft een gestructureerd testproces.
3. TMap bevat een complete gereedschapkist.
4. TMap is een adaptieve testmethode.

De eerstgenoemde essentie is direct te relateren aan het feit dat de business case van IT steeds belangrijker wordt voor organisaties. De aanpak volgens BDTM geeft binnen TMap invulling aan dit feit en is daarmee dan ook te zien als de ‘rode draad’ van het gestructureerde TMap testproces (essentie 2). Bij de beschrijving van het testproces wordt gebruik gemaakt van het TMap faseringsmodel. Hiernaast moeten, voor het goed kunnen uitvoeren van het testproces, diverse zaken op het gebied van infrastructuur, technieken en organisatie worden ingericht. TMap geeft hierover veel praktisch toepasbare informatie in de vorm van onder andere voorbeelden, checklists, techniekbeschrijvingen, procedures, testorganisatiestructuren, testomgevingen en testtools (essentie 3). Verder is TMap flexibel opgezet zodat het toepasbaar is in diverse systeemontwikkelsituaties: bij zowel nieuwbouw als onderhoud van een systeem, bij een zelfontwikkeld systeem of aangeschaft pakket en bij uitbesteding van (delen van) het testen. Met andere woorden: TMap is een adaptieve methode (essentie 4).

In figuur 3.1 “TMap essentiemodel” symboliseert de linker driehoek BDTM, de onderste driehoek de gereedschapkist, de parallellogram het gestructureerde testproces en de ‘cirkel’ de adaptiviteit van TMap.



Figuur 3.1: TMap essentiemodel.

3.1 Business driven toegelicht

De kern van testen is dat er tests worden uitgevoerd aan de hand van bijvoorbeeld testgevallen of checklists. Maar wat voor tests zijn dit? Om het testen zinvol te laten zijn, moeten de tests zijn gericht op het testen van *die* kenmerken en onderdelen van een testobject waarvoor men een risico ziet als deze later in productie onvoldoende goed werken. Dit betekent dat er, voordat er met de testuitvoering kan worden gestart, blijkbaar al diverse afwegingen hebben plaatsgevonden. Er is dus al nagedacht over wat van welke onderdelen van het testobject niet, wel, hoe en met welke

dekking moet worden getest. Waar wordt dat dan door bepaald? Waarom worden niet alle onderdelen van het testobject zo zwaar mogelijk getest? Als een organisatie over onbeperkte middelen zou beschikken dan zou er inderdaad voor gekozen kunnen worden alles zo zwaar mogelijk te testen. In de praktijk blijkt een organisatie natuurlijk nooit over zoveel middelen te beschikken om dat daadwerkelijk zo uit te kunnen voeren. Dit betekent dat er keuzes moeten worden gemaakt in wat en hoe zwaar moet worden getest. Deze keuzes zijn afhankelijk van de risico's die een organisatie denkt te lopen, van de beschikbare hoeveelheid geld en tijd en van het resultaat wat de organisatie wil bereiken. Het feit dat de keuzes gebaseerd zijn op risico's, resultaat, tijd en kosten wordt business driven genoemd en is de basis voor de BDTM aanpak. Voor het kunnen begrijpen en toepassen van de BDTM aanpak wordt eerst het begrip "business case" en daarna BDTM zelf toegelicht.

Business case als bepalende factor

IT projecten moeten steeds meer puur economisch worden bekeken. De theorie van IT-governance laat projecten sturen op een viertal aspecten: resultaat, risico, tijd en kosten. Zo kan het voor een organisatie een aantrekkelijkere investering zijn om een risicovol project te starten, dat in potentie een hoog resultaat geeft, dan een project met nauwelijks risico's maar waarvan de baten maar weinig uitstijgen boven de kosten.

Als het goed is, ligt aan de basis van een IT project een business case ten grondslag. Er zijn verschillende definities van het begrip business case in omloop. Bijvoorbeeld onderstaande business case definitie voor projecten volgens [PRINCE2, 2002].

Definitie

De business case geeft de rechtvaardiging voor het project weer en geeft antwoord op de vragen: *waarom* doen we dit project, *welke* investeringen zijn hiervoor nodig, *wat* wil de opdrachtgever met het resultaat bereiken?

In vastgestelde termijnen wordt tijdens het project de business case getoetst om er zeker van te zijn dat de uiteindelijke resultaten valide blijven voor de opdrachtgever.

TMap ondersteunt genoemde rechtvaardiging van IT en vertaalt deze door naar het testen. In TMap wordt er vanuit gegaan dat een projectaanpak, die is gebaseerd op een business case, voldoet aan de volgende kenmerken:

- De aanpak is gericht op het behalen van een vooraf gedefinieerd resultaat.
- Het totale project om dit resultaat te behalen, wordt binnen de beschikbare (doorloop)tijd gerealiseerd.
- Het project om dit resultaat te behalen, wordt gerealiseerd tegen kosten die in balans zijn met de baten die de organisatie wil behalen.
- De risico's bij in-productiename zijn bekend en zo klein mogelijk. Dit alles binnen de kaders die gesteld worden door de bovenstaande kenmerken.

In deze kenmerken zijn genoemde vier IT-governance aspecten terug te vinden. Voor het succesvol kunnen uitvoeren van een project is het van belang dat het testproces met de business case in overeenstemming wordt gebracht. De relatie tussen de business case en het testproces wordt via de business driven testmanagement aanpak gelegd. Anders gezegd: met deze aanpak kan een 'vertaling' van de business case kenmerken naar het testproces worden gemaakt.

Kenmerken van de business driven testmanagement aanpak

Vaak blijken testplannen en -rapportages de opdrachtgever niet aan te spreken. De oorzaak hiervan ligt in het verleden toen de tester vrijwel altijd vanuit de IT redeneerde. Het testproces was daarbij intern gericht en vol van test- en IT-vakjargon. Dit maakte het lastig met een niet-IT opdrachtgever zoals een gebruikersafdeling te communiceren, terwijl dat wel noodzakelijk is.

In TMap wordt, door de aanpak volgens business driven testmanagement[1], expliciet aandacht besteed aan communicatie. BDTM hanteert hierbij als uitgangspunt dat de gekozen testaanpak de opdrachtgever in staat moet stellen om het testproces te kunnen sturen en de testaanpak (mede) te kunnen bepalen. Hiermee krijgt testen een economischer karakter. Vanuit het testproces wordt de benodigde informatie geleverd om dit mogelijk te maken.

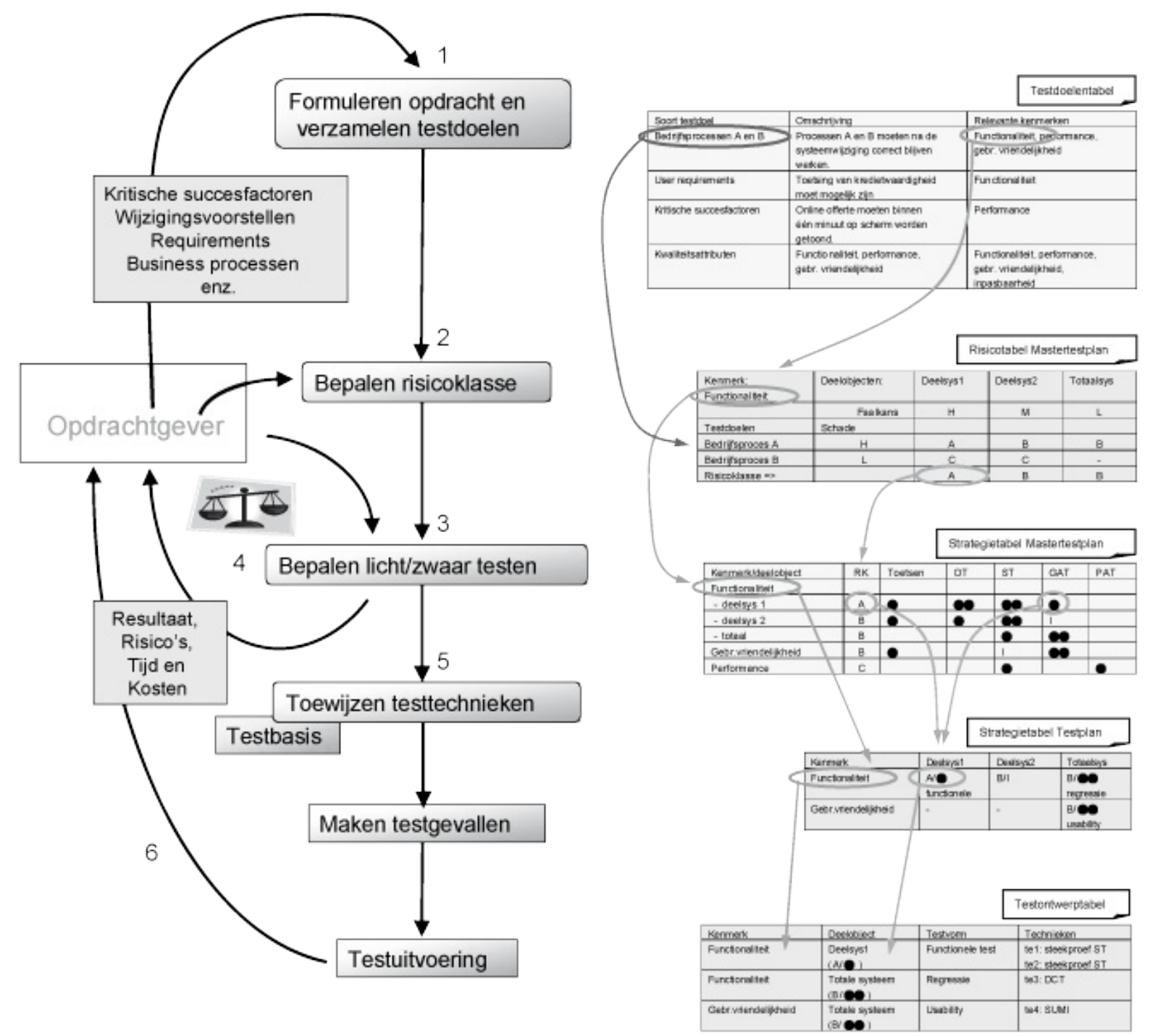
BDTM heeft de volgende specifieke kenmerken:

- De totale testinspanning is gerelateerd aan de risico's van het te testen systeem voor de organisatie. De inzet van mensen, middelen en budget is daardoor gericht op die delen van het systeem die voor de organisatie het belangrijkst zijn. Binnen TMap zijn de teststrategie, in combinatie met de begroting, middelen om de testinspanning over systeemdelen te verdelen en zo inzicht te verschaffen in de mate waarin risico's al dan niet worden afgedekt.
- De begroting en planning voor het testproces zijn gerelateerd aan de opgestelde teststrategie. Als er wijzigingen worden doorgevoerd die gevolgen hebben voor de zwaarte waarmee de verschillende systeemdelen of systemen moeten worden getest, wordt dit meteen vertaald in een wijziging in de begroting en/of planning. De organisatie heeft daardoor steeds een goed beeld van het benodigde budget, de doorlooptijd en de relatie met de teststrategie.
- Op verschillende momenten in het testtraject wordt de opdrachtgever betrokken bij het maken van keuzes. Voordeel hiervan is dat het testproces zo goed mogelijk aansluit bij de eisen en wensen en daarmee bij de verwachtingen van de organisatie. Bovendien biedt BDTM handvatten om de gevolgen van te maken en gemaakte keuzes expliciet zichtbaar te maken.

De stappen van de business driven testmanagement aanpak

Om de BDTM aanpak goed te kunnen begrijpen, is het van belang om het einddoel niet uit het oog te verliezen. Namelijk, het kunnen geven van een kwaliteitsoordeel en risicoadvies over het systeem. Omdat nooit alles kan worden getest, kan enkel tot een goed oordeel worden gekomen door het maken van een zo goed mogelijke verdeling van de testinspanning, in termen van tijd en geld, over delen en kenmerken van het te testen systeem. De stappen van BDTM zijn hierop gericht (zie figuur 3.2):

1. Formuleren opdracht en verzamelen testdoelen.
   De testmanager formuleert in samenspraak met de opdrachtgever de opdracht, waarbij rekening wordt gehouden met de vier BDTM-aspecten: resultaat, risico, tijd en kosten.
   Voor het vaststellen wat de gewenste resultaten van het testen voor de opdrachtgever moeten zijn, verzamelt de testmanager de testdoelen. Een testdoel is een voor de opdrachtgever en andere acceptanten relevant doel voor het testen, vaak geformuleerd in termen van door IT ondersteunde business processen, gerealiseerde user requirements of use cases, kritische succesfactoren, wijzigingsvoorstellen of benoemde (af te dekken) risico's.
2. Bepalen risicoklasse per combinatie van kenmerk en deelobject.
   Als er sprake is van *meerdere* testsoorten wordt in een plan over alle testsoorten heen bepaald welke testsoorten moeten worden ingericht (mastertestplan). Vaak wordt dan al op basis van een productrisicoanalyse[2] bepaald *wat* er moet worden getest (deelobjecten) en *waar naar* moet worden gekeken (kenmerken).



Figuur 3.2: BDTM-stappen.

Als er sprake is van slechts *één* testsoort of als er op mastertestplanniveau geen of een globale productrisicoanalyse heeft plaatsgevonden, wordt binnen de betreffende testsoort een (eventueel aanvullende) productrisicoanalyse uitgevoerd. Het uiteindelijke resultaat (in één keer, of na één of meer aanvullende analyses) is een risicotabel waarin per deelobject een, aan de testdoelen en het betreffende kenmerk gerelateerde, risicoklasse is vastgelegd ("Risicotabel Mastertestplan"). Vervolgens wordt in een tabel per combinatie van kenmerk/deelobject en testsoort richting gegeven aan de relatieve diepgang waar mee moet worden getest ("Strategietabel mastertestplan").

Nu ontstaat er een *iteratief traject*:

3. Bepalen of een combinatie van kenmerk en deelobject licht of zwaar moet worden getest.
   Voor het bepalen hoe licht of hoe zwaar er moet worden getest, wordt de in de vorige stap per deelobject vastgestelde risicoklasse als uitgangspunt gehanteerd. Hierbij geldt in eerste instantie: hoe meer risico, hoe zwaarder er moet worden getest. Het resultaat hiervan wordt vastgelegd in een strategietabel per testsoort ("Strategietabel testplan").
4. Vervolgens wordt de test op hoofdlijnen begroot en in een planning uit gezet. Dit wordt afgestemd met opdrachtgever en andere betrokkenen en afhankelijk van hun oordeel mogelijk herzien. In dat geval worden stappen 3 en 4 opnieuw doorlopen. De opdrachtgever heeft hierdoor nadrukkelijk grip op het testproces en kan deze sturen in de balans resultaat en risico versus tijd en kosten.

Einde iteratie

5. Toewijzen testtechnieken aan de combinaties van kenmerk en deelobject.
   Als opdrachtgever en de betrokkenen het eens zijn met de begroting en de planning, vult de testmanager een "Testontwerptabel" in. Hierin zijn de beslissingen over zwaar en minder zwaar testen vertaald naar concrete uitspraken over welke dekking nagestreefd wordt. Vervolgens wijst hij testtechnieken toe aan de combinaties van kenmerk en deelobject. Hierbij wordt rekening gehouden met onder andere de beschikbare testbasis. Met deze technieken worden in een later stadium de testgevallen (en/of checklists) ontworpen en uitgevoerd. Nu start het primaire testproces.
6. De testmanager geeft gedurende het gehele testproces aan de opdracht gever en andere betrokkenen voldoende inzicht in én sturingsmogelijkheden over:
   - de voortgang van het testproces;
   - de kwaliteit en risico's van het testobject;
   - de kwaliteit van het testproces.

Samenvattend zijn de voordelen van de BDTM aanpak:

- een door de opdrachtgever stuurbaar proces; de testmanager communiceert en rapporteert in de terminologie van de opdrachtgever met informatie die zinvol is in de context van de opdrachtgever. Bijvoorbeeld door te rapporten in termen van testdoelen (bijvoorbeeld bedrijfsprocessen) in plaats van deelobjecten kenmerken;
- op mastertestplanniveau kan net zover worden gedetailleerd als gewenst of mogelijk is, hierdoor kan mogelijk het uitvoeren van een productrisicoanalyse of het opstellen van een teststrategie voor de afzonderlijke testsoorten met minder inspanning worden gedaan of zelfs worden overgeslagen (toelichting op mastertestplan volgt in volgende paragraaf).

3.2 Gestructureerd testproces

In deze paragraaf worden de fasering en de activiteiten van volgende TMap processen beschreven:

- mastertestplan, managen van het totale testproces;
- acceptatie- en systeemtesten;
- ontwikkeltesten.

Mastertestplan en andere TMap processen

Wanneer de testmanager van iedere testsoort met de directe afnemers bepaalt wat getest gaat worden, is de kans groot dat in het totaalbeeld van het testen bepaalde zaken onnodig dubbel getest worden. Een andere mogelijkheid is dat juist bepaalde aspecten tussen wal en schip vallen. De werkwijze moet dan ook andersom zijn. Passend binnen de gehanteerde ontwikkel- of onderhoudsaanpak maakt een testmanager in overleg met de opdrachtgever en andere betrokkenen het totaaloverzicht van de verdeling over de testsoorten van wat, wanneer en hoe zwaar moet worden getest. Het streven is hierbij om de belangrijkste fouten zo vroeg en goedkoop mogelijk te ontdekken. Deze afstemming wordt vastgelegd in het zogenaamde mastertestplan (MTP). Dit plan vormt de basis voor de testplannen van de afzonderlijke testsoorten. Naast deze inhoudelijke afstemming zijn andere redenen om af te stemmen: het hanteren van uniformiteit in processen (bijvoorbeeld de bevindingenprocedure en het beheer van de testware), beschikbaarheid en beheer van de testomgeving en -tools, en optimale verdeling van de resources (zowel mensen als middelen) over de testsoorten heen.

Dit betekent dat naast testsoorten als acceptatie-, systeem- en ontwikkeltesten ook het mastertestplan een belangrijke rol binnen TMap speelt. Zowel voor het mastertestplan als voor de testsoorten geldt dat het belangrijk is om voor het maken van plannen, het voorbereiden, uitvoeren en beheren van activiteiten een goed proces in te richten.

Hoewel de doelen van de testsoorten acceptatie- en systeemtest verschillen, worden deze testsoorten niet apart, maar als één proces beschreven. Hier is voor gekozen omdat de activiteiten in beide testsoorten vrijwel hetzelfde zijn en aparte procesbeschrijvingen daardoor (te)veel overlap zouden vertonen.

Naast deze processen is nog het proces "Ondersteunende processen" gedefinieerd, omdat het efficiënter is bepaalde aspecten/ondersteuning centraal te organiseren in plaats van per project. Dit betreft ondersteunende processen voor de volgende onderwerpen:

- testbeleid
- permanente testorganisatie

- testomgevingen
- testtools
- testprofessional.

De ondersteunende processen worden op daarvoor relevante plaatsen, als onderdeel van de complete gereedschapkist, toegelicht (zie paragraaf 3.3).

3.2.1 Proces mastertestplan, managen van het totale testproces

Het mastertestplan geeft inzicht in de diverse toe te passen test- en toetssoorten zodanig dat er een optimalisatie plaatsvindt van het totale testproces en is sturend voor de onderliggende testsoorten.

Het proces “Mastertestplan, managen van het totale testproces” is verdeeld in twee fasen: de fase Planning en de fase Beheer.

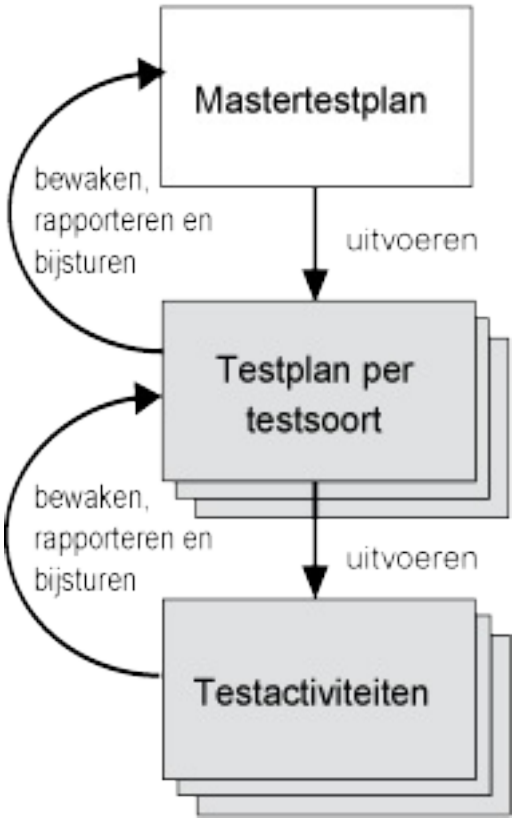
Fase Planning van het totale testproces

De opsteller van het MTP, vaak de testmanager, formuleert in samenspraak met de opdrachtgever de opdracht, rekening houdend met de vier BDTM-aspecten resultaat, risico’s, tijd en kosten. Vervolgens gaat de testmanager zich inwerken in het aankomende traject door diverse gesprekken met belanghebbenden te voeren en andere informatiebronnen zoals documentatie te raadplegen. Parallel hieraan vult de testmanager de opdracht verder in en bepaalt samen met opdrachtgever het beschouwingsgebied. In deze fase worden de eerste vier stappen van BDTM doorlopen: uitvoeren van een PRA, opstellen van een teststrategie, begroting en planning (zie ook figuur “3.2 BDTM-stappen”).

Verdere activiteiten in de planvorming zijn dat de testmanager de producten definieert die de testsoorten moeten opleveren en een voorstel doet hoe de testorganisatie eruit moet gaan zien, op centraal niveau en globaal per testsoort. De testmanager stemt de infrastructuurbehoeften van de testsoorten onderling af om de vaak schaarse testinfrastructuur zo efficiënt mogelijk in te kunnen zetten. Testbeheer kan ook deels al op MTP-niveau worden ingericht. Dit kan zowel door centrale procedures en standaards voor beheer op te stellen als door het centraal beheren van bepaalde zaken. Beide mogelijkheden hebben tot doel te voorkomen dat het wiel in de afzonderlijke testsoorten nogmaals uitgevonden gaat worden. De belangrijkste risico’s, die het testproces bedreigen, worden benoemd en er worden mogelijke maatregelen voorgesteld om deze risico’s te beheersen. Als laatste stap laat de testmanager het MTP goedkeuren door de opdrachtgever.

Fase Beheer van het totale testproces

Het doel van deze activiteit is het op overkoepelend niveau beheren van het testproces, de infrastructuur en de testproducten om voortdurend inzicht te kunnen bieden in de voortgang en kwaliteit van het totale testproces en de kwaliteit van het testobject. Conform de in het testplan vastgelegde frequentie en vorm wordt gerapporteerd over de kwaliteit van het testobject en over de voortgang en kwaliteit van het testproces. Vanaf de eerste testactiviteiten ontstaat bij de testers een beeld over die kwaliteit. Het is belangrijk dat gedurende alle fasen van het testproces hierover wordt gerapporteerd. Aan de opdrachtgever wordt periodiek, en op verzoek ad hoc, gerapporteerd in welke conditie het systeem verkeert. Dit rapporteren en bijsturen is een essentieel onderdeel van de BDTM aanpak (BDTM-stap 6) en gebeurt op zowel mastertestplan als testsoortniveau (figuur 3.3 “Beheer en testprocessen”).



Figuur 3.3: Beheer en testprocessen.

3.2.2 Proces acceptatie- en systeemtesten

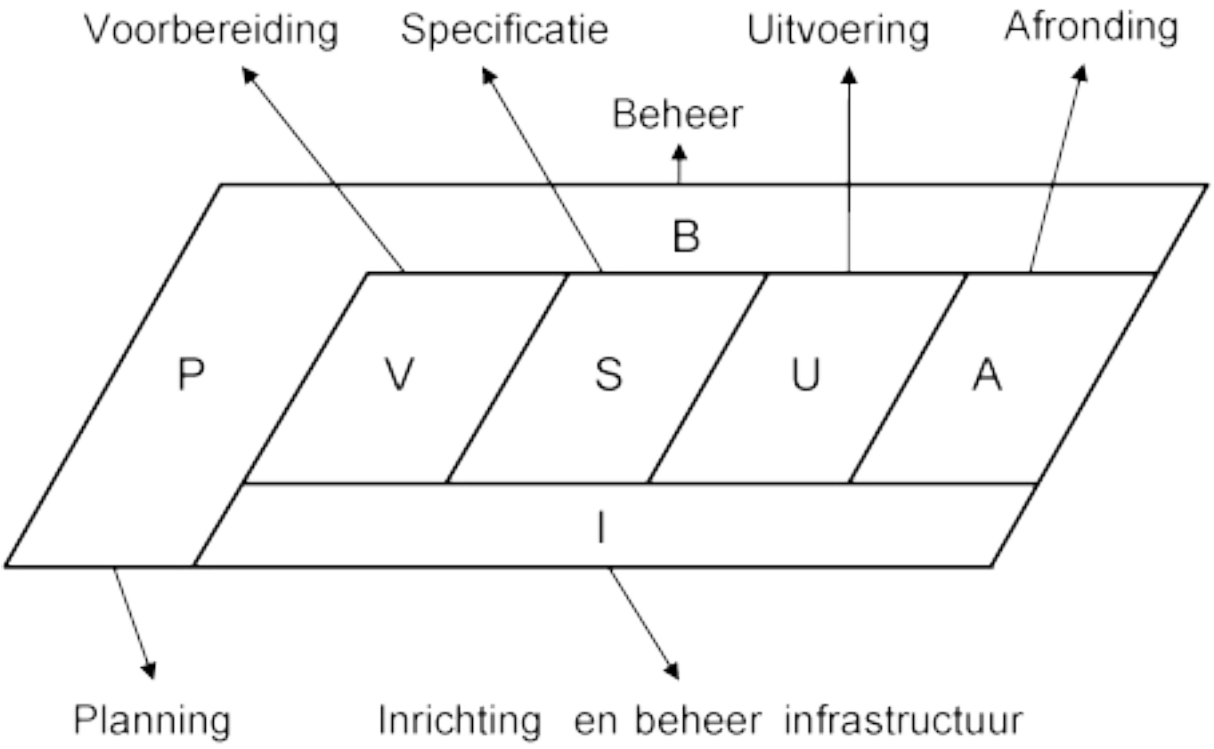
De acceptatietest en de systeemtest worden als op zichzelf staande en te organiseren processen beschouwd. Ze hebben een eigen testplan, een eigen budget en vaak ook een eigen testomgeving waarop de test wordt uitgevoerd.

Het zijn processen die parallel lopen aan het ontwikkelproces en die dienen te starten tijdens het opstellen van de functionele specificaties. Zowel bij het opstellen van het testplan als bij het uitvoeren van de overige activiteiten van het testproces wordt het TMap faseringsmodel gebruikt.

Faseringsmodel

Een testproces bestaat net als een systeemontwikkelproces uit een aantal verschillende activiteiten. Om de verschillende activiteiten en hun onderlinge volgorde en afhankelijkheden overzichtelijk in kaart te brengen, is een testfaseringsmodel nodig. Het faseringmodel is een generiek model. Het is toepasbaar voor alle testsoorten en testvormen, en parallel aan de faseringsmodellen voor systeemontwikkeling te hanteren. In het faseringsmodel van TMap zijn de testactiviteiten verdeeld over een zevental fasen: Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding (zie figuur 3.4 “TMap faseringsmodel”). Elke fase is vervolgens onderverdeeld in activiteiten.

Door gebruik te maken van een testfaseringsmodel wordt het overzicht behouden tijdens het testproces. Door te noteren *wat*, *wanneer*, *hoe*, *waar(mee)*, door *wie*, enzovoort moet gebeuren in het stramien van het proces worden vanzelf de claims op en de relaties met andere aspecten zoals technieken, infrastructuur en organisatie gelegd.



Figuur 3.4: TMap faseringsmodel.

Het kritieke pad en de vorm van het faseringsmodel

Als we het testproces vergelijken met een ijsberg dan zou slechts de fase Uitvoering ‘zichtbaar’ moeten zijn. Dit betekent dat alleen de fase Uitvoering zich op het ‘kritieke pad’ van een project mag bevinden. Alle activiteiten uit de andere fasen kunnen, óf daarvoor, óf daarna plaatsvinden.

De vorm van het faseringsmodel (parallellogram) geeft aan dat de testfasen niet strikt sequentieel hoeven te worden uitgevoerd.

Testfaseringsmodel relaties

De relatie tussen de TMap testfasering en de systeemontwikkelfasering is afhankelijk van de gebruikte systeemontwikkelmethode en de desbetreffende testsoort. Hierbij zijn wel twee ‘vaste’ relaties aan te geven. De start van de fase Voorbereiding heeft een relatie met het moment van beschikbaar zijn van de testbasis en de start van de fase Uitvoering heeft een relatie met het moment van beschikbaar zijn van het testobject.

Fase Planning

De uit te voeren activiteiten in de fase Planning leggen de basis voor een beheersbaar en kwalitatief goed testproces. Het is daarom van belang deze fase zo snel mogelijk te starten. De planningsfase is een belangrijke testfase. Hij wordt vrijwel altijd onderschat. Vaak zijn in een mastertestplan al, op globaal niveau, de kaders gezet voor een bepaalde testsoort. In dat geval vindt in deze fase de detailinvulling plaats.

Nadat de testopdracht is gefixeerd, wordt globaal kennis gemaakt met de testbasis, de materie en de organisatie (van het project). Het is onmogelijk het systeem volledig te testen. Geen enkele organisatie heeft daar tijd en geld voor. Daarom wordt via een proces van risicoanalyse de teststrategie, de begroting en de planning bepaald (BDTM-stappen 1 t/m 4). Een en ander wordt uiteraard nadrukkelijk afgestemd met de opdrachtgever. Vervolgens wordt vastgesteld welke testtechnieken moeten worden gebruikt (BDTM-stap 5). Het doel is het realiseren van de best haalbare dekking op de juiste plaats binnen de gestelde BDTM kaders. Daarnaast wordt een aanzet gegeven tot de inrichting van de testorganisatie en de testinfrastructuur. Deze activiteiten worden in het begin van het testproces uitgevoerd en vastgelegd in het testplan voor de desbetreffende testsoort.

Fase Beheer

Het primaire testproces zal zelden volgens plan worden uitgevoerd. Dus ook de uitvoering van dit testplan zal bewaakt en eventueel bijgestuurd moeten worden. Dit gebeurt in de fase Beheer. Het doel van de activiteiten in deze fase is het op optimale wijze beheersen van en rapporteren over het testproces, zodanig dat de opdrachtgever voldoende inzicht in én sturingsmogelijkheden over de voortgang en kwaliteit van het testproces en de kwaliteit van het testobject krijgt.

De testmanager en/of de beheerder beheren het testproces, de infrastructuur en de testproducten. Op basis van deze gegevens analyseert de testmanager mogelijke trends. Tevens houdt de testmanager bijzonder goed voeling met wat de ontwikkelingen zijn buiten het testen, zoals vertraging bij de ontwikkelaars, komende grote wijzigingsvoorstellen en projectbijsturing. Indien nodig stelt de testmanager de opdrachtgever bepaalde sturingsmaatregelen voor.

Informatie is het belangrijkste product van testen. Hiertoe verzorgt de testmanager verschillende soorten rapportages aan de diverse doelgroepen, rekening houdend met de BDTM elementen resultaat, risico's, tijd en kosten (BDTM-stap 6).

Fase Inrichting en beheer infrastructuur

De fase Inrichting en beheer infrastructuur heeft als doel om zorg te dragen voor de benodigde testinfrastructuur en middelen. Hierbij wordt onderscheid gemaakt tussen testomgevingen, testtools en werkplekken.

Het inrichten en beheren van infrastructuur is een specifieke expertise. Het is iets waar testers over het algemeen beperkte kennis van hebben, maar waar ze wel heel erg van afhankelijk zijn. Zonder infrastructuur is er geen test mogelijk. Alle verantwoordelijkheden rond het inrichten en beheren van infrastructuur zijn daarom vaak bij een aparte beheerafdeling belegd. Bij een testtraject zal dus nauw met deze andere, eventueel externe, partijen moeten worden samengewerkt. Dit betekent dat testmanagers terecht komen in een situatie dat ze niet de bevoegdheden hebben rond de inrichting en beheer van de infrastructuur, maar daar wel van afhankelijk zijn. Daarmee is de zorg voor de inrichting en beheer van de infrastructuur een belangrijk aandachtsgebied voor de testmanager. Om daar de focus gedurende de test op te houden, is het een aparte fase binnen het TMap faseringsmodel. Het is een fase die parallel loopt aan de fasen Voorbereiding, Specificatie, Uitvoering en Afronding. Voor sommige Inrichting en beheer infrastructuur activiteiten bestaan afhankelijkheden met activiteiten uit de andere TMap testfasen.

Fase Voorbereiding

In de fase Voorbereiding wordt de detailintake van de testbasis uitgevoerd.

Het einddoel van deze fase is het kunnen beschikken over een, met de opdrachtgever van de test overeengekomen, testbasis die voldoende van kwaliteit is voor het ontwerpen van de tests.

Hiernaast werkt een vroegtijdige intake van de testbasis kwaliteitsverhogend en worden potentieel kostbare fouten voorkomen. Het ontwikkelteam gaat immers op basis van systeemdocumentatie (is onderdeel van de testbasis) aan de slag om het nieuwe informatiesysteem te ontwikkelen. In deze documentatie kunnen fouten voorkomen die bij het niet tijdig ontdekken ervan veel en vaak kostbaar correctiewerk kunnen veroorzaken. Hoe eerder een fout in een ontwikkelproces wordt gevonden, hoe eenvoudiger (en goedkoper) een fout kan worden hersteld

Fase Specificatie

Tijdens de fase Specificatie worden de benodigde tests en uitgangssituatie(s) gespecificeerd. Het doel is zoveel mogelijk voorbereid te hebben om de testuitvoering zo snel mogelijk te laten verlopen wanneer de ontwikkelaars het testobject opleveren. Deze fase start als de detailintake van de testbasis met succes is afgerond. De testspecificatie loopt parallel aan en ook in de luwte van de realisatie van de software.

Fase Uitvoering

Het doel van de fase Uitvoering is om inzicht te krijgen in de kwaliteit van het testobject door het uitvoeren van de afgesproken tests.

De daadwerkelijke uitvoering van de test begint op het moment dat het testobject, of een afzonderlijk testbaar deel van het testobject, wordt opgeleverd. Eerst wordt het testobject gecontroleerd op volledigheid. Hierna wordt het in de testomgeving geïnstalleerd om te kunnen beoordelen of het geheel naar behoren functioneert. Dit gebeurt door het uitvoeren van een eerste test, de zogenaamde pretest. In deze test wordt globaal getest, met als doel te onderzoeken of het

te testen informatiesysteem, in samenhang met de testinfrastructuur, van voldoende kwaliteit is om uitgebreid getest te kunnen worden. Als dit het geval is, wordt de centrale uitgangssituatie klaargezet. De test kan worden uitgevoerd op basis van de testscripts die in de fase Specificatie zijn opgesteld. In dat geval moet eerst de uitgangssituatie, voor de uit te voeren testscripts, worden klaargezet. Tijdens de uitvoering worden de testresultaten gecontroleerd. De verschillen tussen het voorspelde en het werkelijke resultaat worden, vaak in de vorm van een bevindingrapport, geregistreerd.

Fase Afronding

Met de gestructureerde testaanpak van TMap kan veel winst worden behaald in de herhaalbaarheid van het proces. Hierdoor kunnen producten, mits ze voldoen aan bepaalde eisen, weer hergebruikt worden in een volgende test. Dit kan er dan voor zorgen dat bepaalde activiteiten sneller kunnen verlopen. Producten kunnen tastbare zaken als testgevallen of testomgevingen zijn (testware), maar ook niet-tastbare zaken als ervaringen (procesevaluatie) worden hiertoe gerekend.

Bij het conserveren van de testware wordt een selectie gemaakt uit de vaak grote hoeveelheid testware. Onder testware worden onder andere de testgevallen, testscripts en de beschrijving van de testinfrastructuur verstaan. Tijdens het testproces is getracht de testgevallen in overeenstemming te houden met de testbasis en het ontwikkelde systeem. Als dit niet (geheel) is gelukt, worden de geselecteerde testgevallen in de fase Afronding eerst geactualiseerd, voordat de testware wordt geconserveerd. Het voordeel van het op deze manier conserveren van testware is, dat bij systeemwijzigingen de geconserveerde testware met een beperkte inspanning weer actueel kan worden gemaakt om bijvoorbeeld een (regressie)test uit te kunnen voeren. Er hoeft dus niet een compleet nieuwe test te worden ontworpen.

Verder wordt in deze fase het testproces geëvalueerd. Het doel hiervan is om te leren van de opgedane ervaringen en om deze leerpunten toe te passen in een eventuele nieuwe test. Ook dient dit als input voor het eindrapport, die door de testmanager in de fase Beheer wordt opgesteld.

3.2.3 Proces ontwikkeltesten

Onder ontwikkeltesten wordt verstaan het testen met gebruikmaking van kennis van de technische implementatie van het systeem. Dit begint met het testen van de eerste/kleinste onderdelen van het systeem: routines, units, programma’s, modules, objecten, enzovoort. Nadat is vastgesteld dat de meest elementaire delen van het systeem van voldoende kwaliteit zijn, worden vervolgens de grotere delen van het systeem integraal getest. Hierbij ligt de nadruk op de gegevensdoorvoer en de interfacewerking tussen, bijvoorbeeld, de units tot op deelsysteemniveau.

Plaats van het ontwikkeltesten

De ontwikkeltests maken een onlosmakelijk deel uit van de ontwikkelactiviteiten die worden uitgevoerd door de ontwikkelaar. Ze worden niet georganiseerd als een zelfstandig proces met een onafhankelijk team. Ondanks dat kunnen voor het ontwikkeltestproces wel een aantal verschillende activiteiten, met hun onderlinge volgorde en afhankelijkheden, worden geïdentificeerd en met behulp van het TMap faseringsmodel worden beschreven. De detailinvulling hiervan kan per project of organisatie variëren en is bijvoorbeeld afhankelijk van de gebruikte ontwikkelmethode en het aanwezig zijn van bepaalde kwaliteitsmaatregelen.

Een belangrijke kwaliteitsmaatregel is het concept van de afgesproken kwaliteit. Hiervoor moeten, bij de planvorming voor het inrichten van het ontwikkeltesten, de verwachtingen van de opdrachtgever ten aanzien van het vakmanschap en productkwaliteitkwaliteit expliciet worden maken. Voorbeelden van andere kwaliteitsmaatregelen zijn: test driven development, pair programming, code review, continuous integration en de applicatie integrator aanpak.

Verschillen tussen ontwikkeltesten en systeem-/ acceptatietesten

De ontwikkeltest vereist een ‘eigen’ aanpak, die een adequate invulling geeft aan onderstaande verschillen tussen de ontwikkeltest en de systeem-/acceptatietest:

- In tegenstelling tot de systeemtest en de acceptatietest zijn de ontwikkeltests niet te organiseren als een zelfstandig proces met min of meer onafhankelijke teams.
- Bij ontwikkeltesten wordt gebruik gemaakt van kennis van de technische implementatie van het systeem, waardoor andersoortige fouten worden gevonden dan bij systeem- en acceptatietests.

- Bij de ontwikkeltest is de ontdekker van de fouten vaak dezelfde persoon als de oplosser van de fouten.
- De insteek van het ontwikkeltesten is dat alle geconstateerde fouten zijn opgelost vóórdat de software wordt overgedragen.
- Het is het eerste testtraject, wat betekent dat alle fouten nog in het product zitten.
- Bij ontwikkeltesten zijn het meestal de ontwikkelaars zelf die testen.

3.3 Complete gereedschapkist

Het goed kunnen uitvoeren van het gestructureerde testproces wordt door TMap ondersteund door een complete gereedschapkist. Deze gereedschapkist richt zich op de invulling van de onderwerpen:

- Technieken: *hoe* wordt er getest
- Infrastructuur: *waar* en *waarmee* wordt er getest
- Organisatie: *wie* testen er

De diverse ‘gereedschappen’ worden in TMap op het moment dat ze kunnen worden gebruikt in meer detail beschreven. Met deze gereedschapkist beschikt de tester over een groot aantal mogelijkheden om elke testuitdaging succesvol aan te kunnen gaan.

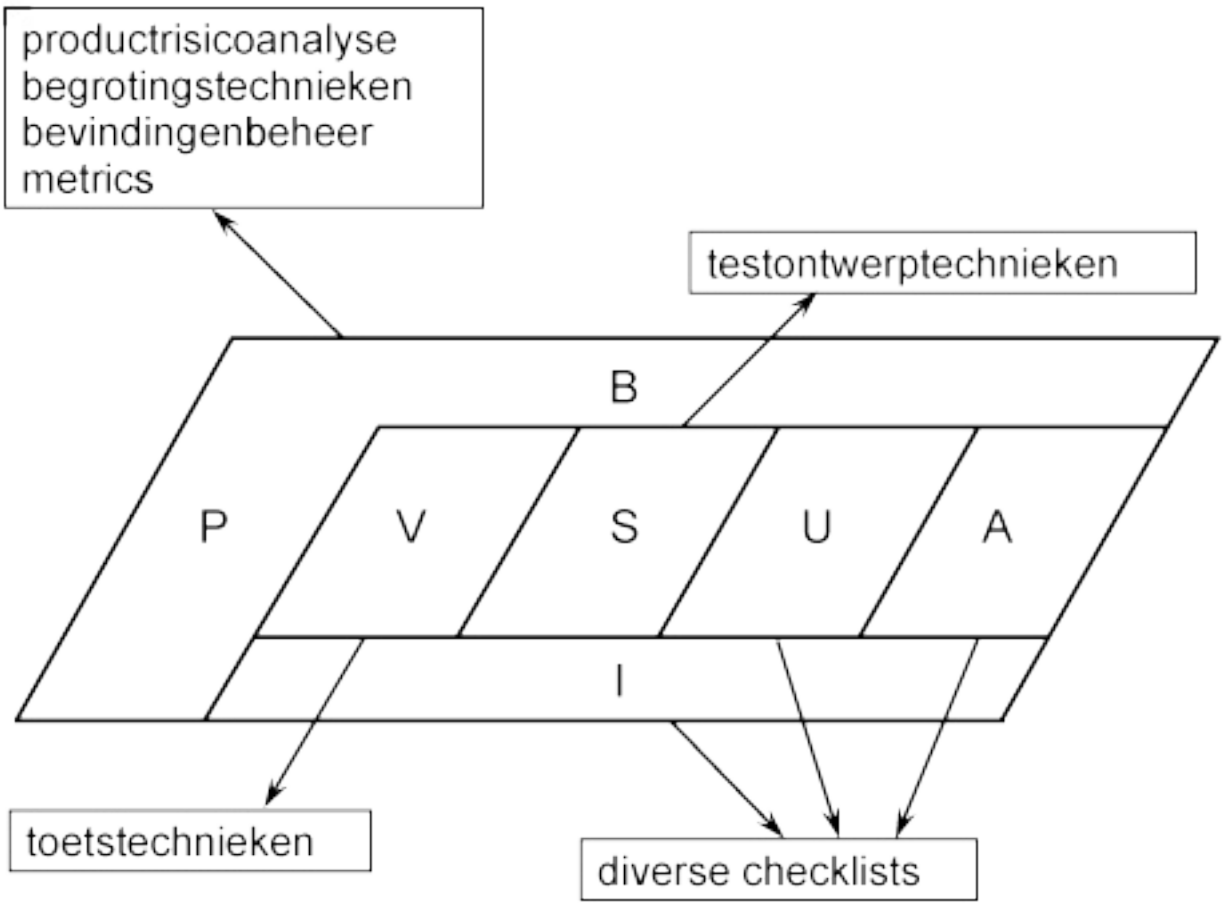
3.3.1 Technieken

In het testproces kan van vele technieken gebruik worden gemaakt. Een testtechniek is een samenstel van acties om op universele wijze een testproduct te produceren.

TMap biedt technieken voor de volgende onderwerpen (zie figuur 3.5):

- begroten van de test
- beheren van bevindingen
- opstellen van metrics
- analyseren van productrisico’s
- ontwerpen van tests
- toetsen van producten.

Hiernaast biedt TMap nog diverse checklists en overzichten, die bij het voorbereiden en of uitvoeren van bepaalde activiteiten als hulpmiddel kunnen worden gebruikt.

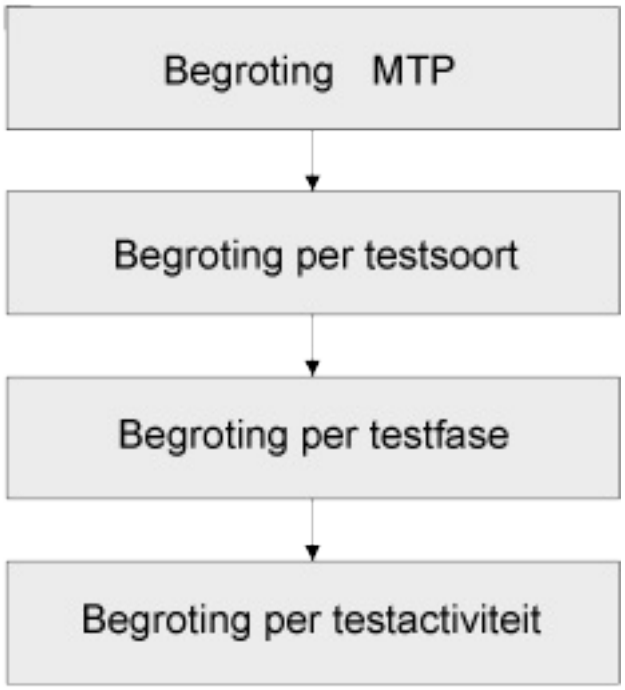


Figuur 3.5: Testtechnieken.

In de volgende paragrafen worden de (groepen) testtechnieken kort gekarakteriseerd.

Begroten van de test

Er zijn een aantal niveaus waarop een begroting kan worden gemaakt. De verschillende niveaus van begroting staan in figuur 3.6.



Figuur 3.6: Niveaus van begroting.

Het opstellen van een begroting bestaat, onafhankelijk van het niveau, uit de volgende generieke stappen:

1. Inventariseer het beschikbare materiaal dat als basis voor het begroten kan dienen.

2. Kies (een aantal) begrotingstechnieken.
3. Stel de uiteindelijke begroting vast.
4. Presenteer de uitkomst.

Met name het kiezen van de begrotingstechnieken is hierbij een stap waarvoor ervaring is vereist. Er kan worden gekozen uit diverse begrotingstechnieken:

- Begroten op basis van verhoudingsgetallen. Hierbij wordt de testinspanning veelal afgemeten aan de ontwikkelinspanning, bijvoorbeeld in percentuele verhoudingen.
- Begroten op basis van testobject omvang.
- Begroten met behulp van een ‘work breakdown structure’.
- Proportioneel begroten op basis van het totale testbudget.
- Begroten op basis van het extrapoleren van ervaringscijfers uit het begin van het testtraject.
- Begroten op basis van omvang en strategie met behulp van TMap’s testpuntanalyse (TPA).

Hiernaast geeft TMap nog een techniek voor het opstellen van een toetsbegroting.

Beheren van bevindingen

Een bevinding is een geconstateerd verschil tussen de verwachting of voorspelling en de feitelijke uitkomst. Hoewel het administreren en bewaken van de bevindingen feitelijk een projectaangelegenheid is en niet specifiek een zaak van de testers, zijn testers hier wel het meest bij betrokken. Een goede administratie moet de levensloop van een bevinding kunnen bewaken en daarnaast diverse overzichten kunnen geven. Deze overzichten worden onder andere gebruikt om gefundeerde kwaliteitsuitspraken te doen.

Opstellen van metrics

Het definiëren, bijhouden en gebruiken van metrics is voor het testproces van belang omdat de testmanager hierdoor een met feiten onderbouwd antwoord kan geven op vragen als:

- Hoe staat het met de kwaliteit van het testobject?
- Hoe zit het met de voortgang van het testproces?

Een gestructureerde aanpak om tot een set van testmetrics te komen kan door het gebruik van de Goal-Question-Metric (GQM) methode.

Naast het geven van een beschrijving van de GQM methode geeft TMap aanwijzingen voor het opzetten van een praktische testmetrics beginset. Verder wordt er een checklist gegeven die behulpzaam kan zijn bij het doen van zowel een uitspraak over de kwaliteit van het te testen object als over de kwaliteit van het testproces.

Analyseren van productrisico’s

De productrisicoanalyse (PRA) is het analyseren van het te testen product met als doel dat testmanager en de verschillende andere belanghebbenden tot een gezamenlijk beeld komen van wat de meer of minder risicovolle kenmerken en onderdelen van het te testen product zijn, zodat de grondigheid van testen hieraan gerelateerd kan worden. De focus bij de PRA ligt op de productrisico’s, oftewel: wat is het risico voor de organisatie wanneer het product niet de verwachte kwaliteit heeft.

Het resultaat van de PRA vormt de basis voor de latere keuze in de strategie voor licht, zwaar of niet testen van een kenmerk (bijvoorbeeld een kwaliteitsattribuut) of deelobject (onderdeel) van het te testen product.

Ontwerpen van tests

Een testontwerptechniek is een gestandaardiseerde werkwijze om vanuit een bepaalde testbasis testgevallen af te leiden die een bepaalde dekking bereiken.

Er zijn diverse voordelen te geven voor het toepassen van testontwerptechnieken en het vastleggen ervan in de testspecificaties:

- Het geeft een onderbouwde invulling van de teststrategie: de afgesproken dekking op de afgesproken plaats.
- Het is een effectievere wijze van fouten opsporen dan met bijvoorbeeld ad hoc testgevallen.
- De tests zijn reproduceerbaar, omdat de volgorde en de inhoud van de testuitvoering in detail beschreven zijn.
- De gestandaardiseerde werkwijze maakt het testproces onafhankelijk van de persoon die de testgevallen specificeert en uitvoert.
- De gestandaardiseerde werkwijze maakt de testspecificaties overdraagbaar en onderhoudbaar.
- Het testproces is beter planbaar en beheersbaar, omdat de processen van testspecificatie en -uitvoering in goed gedefinieerde blokken kunnen worden opgedeeld.

Testontwerptechnieken komen in vele varianten en combinaties voor. De testontwerptechnieken die in TMap beschreven zijn, vormen een gevarieerde set waarmee de meeste organisaties direct aan de slag kunnen. TMap beschrijft de volgende dekkingsvormen en testontwerptechnieken.

- Dekkingsvormen
  paden, beslispunten, equivalentieklassen, pairwise testing, orthogonale arrays, grenswaardenanalyse, CRUD, operational- en load profiles, goed- en foutpaden en checklists
- Testontwerptechnieken
  beslistabeltest, datacombinatietest, elementaire vergelijkingstest, error guessing, exploratory testing, gegevenscyclustest, procescyclustest, real life test, semantische test, syntactische test en use case test.

Toetsen van producten

In TMap worden volgende toetstechnieken beschreven en gebruikt:

- Inspecteren
  Naast het toetsen of de oplossing goed is verwerkt, is een inspectie vooral gericht op het verkrijgen van consensus over de kwaliteit van een product.
- Reviewen
  Een review is vooral gericht op het zoeken naar oplossingsrichtingen op basis van kennis en vaardigheden van de reviewers én op het vinden en corrigeren van fouten.
- Walkthrough
  Het houden van een walkthrough is een werkwijze waarbij de auteur van een bepaald product in een bijeenkomst uitlegt wat de inhoud van een product is. Hierbij zijn verschillende doelen mogelijk:
  – alle deelnemers op hetzelfde uitgangsniveau brengen, bijvoorbeeld als voorbereiding op een review- of inspectieproces;
  – overdracht van informatie, bijvoorbeeld aan ontwikkelaars en testers om hen te helpen bij respectievelijk hun programmerings- en testontwerpwerkzaamheden;
  – aan de deelnemers vragen om aanvullende informatie;
  – de deelnemers laten kiezen uit door de auteur aangedragen alternatieven.

Diverse checklists en overzichten

TMap biedt een grote diversiteit aan checklists, die bij het uitvoeren van bepaalde activiteiten voor de tester een welkome aanvulling zullen zijn. Er zijn bijvoorbeeld checklists die gebruikt kunnen worden als ondersteuning bij de opdrachtoriëntering, bij het bepalen van de testfaciliteiten, bij het vaststellen van de testprojectrisico's, bij het opstellen van de teststrategie, bij de evaluatie van het testproces, bij het afnemen van interviews en bij het bepalen of er genoeg informatie aanwezig is om een bepaalde testontwerptechniek te kunnen gebruiken. Hiernaast biedt TMap nog andere hulpmiddelen, zoals een overzichtsmatrix van de geautomatiseerde tools per TMap activiteit, een testvormenoverzicht en criteria voor het selecteren van een tool.

Genoemde hulpmiddelen en nog veel meer zijn op www.tmap.net te vinden en daar te downloaden.

3.3.2 Infrastructuur

Om tests te kunnen uitvoeren zijn testomgevingen, testtools en werkplekken nodig.

Testomgevingen

Voor het dynamisch testen van een testobject (runnen van software) is een passende testomgeving nodig. Een testomgeving is een samenstelling van onderdelen zoals hard- en software, koppelingen, omgevingsdata, beheertools en processen waarin een test wordt uitgevoerd. Bepalend voor een succesvolle testomgeving is de mate waarin kan worden vastgesteld in hoeverre het testobject aan de gestelde eisen voldoet. De inrichting en samenstelling van een testomgeving zijn dus afhankelijk van het doel van de test. Niettemin is een aantal generieke eisen te formuleren waaraan een testomgeving moet voldoen om een betrouwbare testuitvoering te garanderen. Deze moet naast het representatief, beheersbaar en flexibel zijn, ook de continuïteit van de testuitvoering garanderen.

Het inrichten en beheren van de testomgeving is een expertise waar testers over het algemeen geen kennis van hebben. Daarom is het inrichten en beheren van de testomgeving vaak belegd bij een aparte afdeling, buiten het project.

Testtools

Om de tests bovendien efficiënt te kunnen uitvoeren, zijn hulpmiddelen in de vorm van testtools noodzakelijk. Een testtool is een geautomatiseerd hulpmiddel dat ondersteuning biedt aan één of meer testactiviteiten, zoals planning en beheer, testspecificatie en testuitvoering. Het gebruik van tools kan de volgende voordelen hebben:

- hogere productiviteit
- hogere kwaliteit testen
- hogere arbeidsvreugde
- uitbreiding testmogelijkheden.

De testtools zijn ingedeeld in vier groepen:

- tools voor het plannen en beheren van de test
- tools voor het ontwerpen van de test
- tools voor het uitvoeren van de test
- tools voor het debuggen en analyseren van de code.

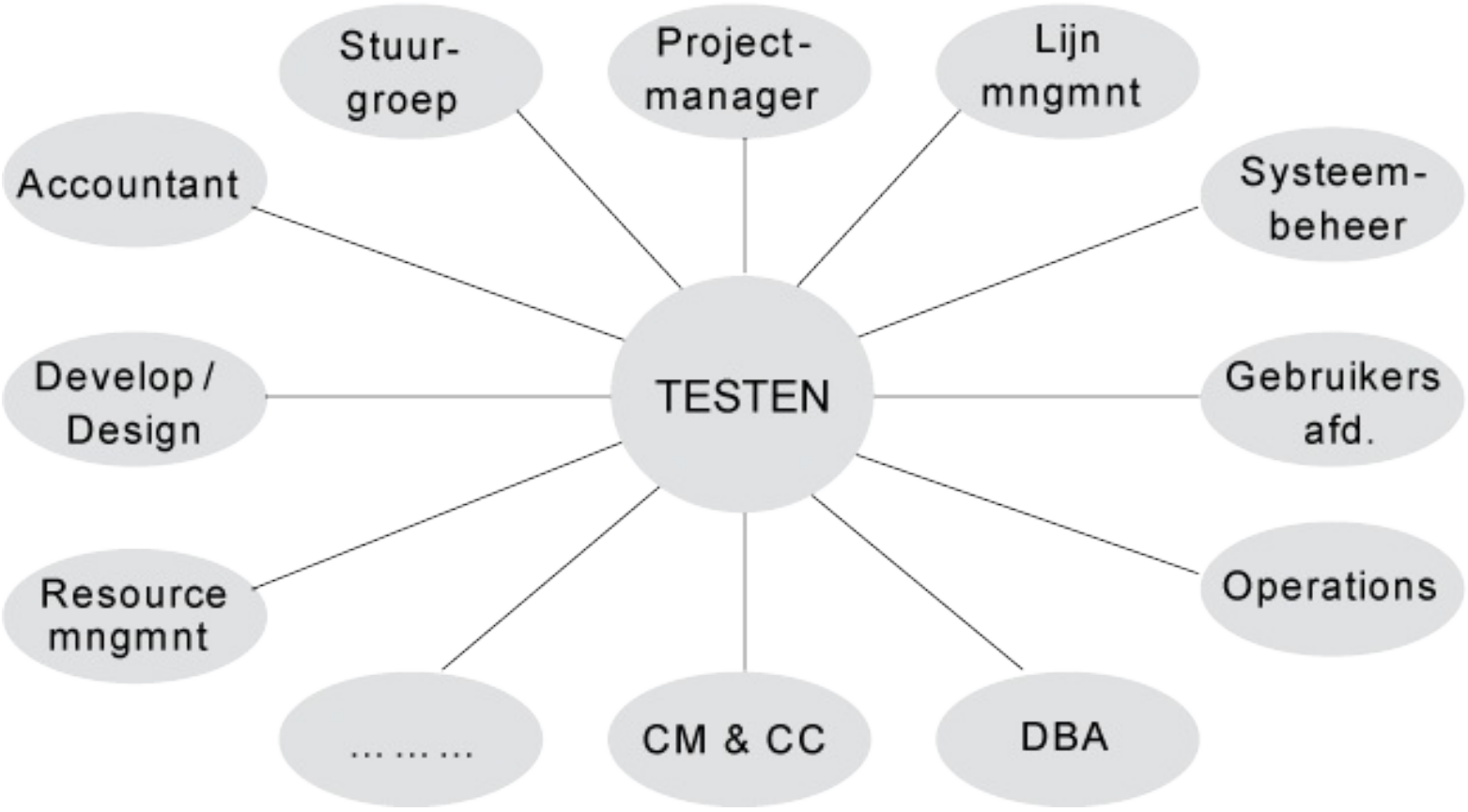
Werkplekken

Een van de aspecten die bij het testen vaak wordt vergeten, is het beschikbaar stellen van een werkplek, waar testers hun werk onder goede omstandigheden effectief en efficiënt kunnen uitvoeren. Het gaat hierbij om kantoorinrichting in de breedste zin van het woord, want ook de testers moeten hun werk onder goede omstandigheden kunnen uitvoeren. De werkplek omvat dus meer dan alleen kantoorruimte en een PC. Ook zaken als bijvoorbeeld toegangspasjes, stroomvoorziening en faciliteiten voor het nuttigen van de lunch moeten geregeld worden. De werkplek voor een tester wijkt zo op het eerste gezicht niet veel af van de reguliere werkplek. Maar schijn bedriegt. Wat getest wordt, is vaak nieuw voor de organisatie en de werkplek. Testers kunnen dan te maken krijgen met de situatie dat hun werkplek nog niet is voorbereid op de nieuwe software. Daarom is het vaak nodig om voor testers aparte autorisaties te regelen. Zo moeten testers bijvoorbeeld de mogelijkheid krijgen om de nieuwe software te installeren op hun lokale PC. Ook voor het gebruik van bepaalde testtools kan dit noodzakelijk zijn.

3.3.3 Organisatie

Elk testproces, waarvan de organisatie van onvoldoende kwaliteit is, loopt uit op een fiasco. De betrokkenheid van veel verschillende disciplines (zie figuur 3.7), de tegenstrijdige belangen, de onvoorspelbaarheid, de ingewikkelde beheertaken, het gebrek aan ervaring(scijfers) en de tijdsdruk maken van de inrichting en de beheersing van de testorganisatie geen gemakkelijke opgave. Enerzijds is er de organisatie binnen het testteam waar ieder zijn taken en verantwoordelijkheden moet krijgen, anderzijds is er de inbedding van het testteam in de projectorganisatie.

Een testorganisatie kan worden gezien als het scheppen van doelmatige verhoudingen tussen testfuncties, testfaciliteiten en testactiviteiten teneinde tijdig een goed kwaliteitsadvies uit te brengen.



Figuur 3.7: De vele verschillende betrokken disciplines.

De organisatie van gestructureerd testen vraagt aandacht op de volgende gebieden:

- testbeleid
- permanente testorganisatie
- testorganisatie in projecten
- testprofessional
- testrollen.

Testbeleid

In het testbeleid staat hoe een organisatie omgaat met de mensen, middelen en methoden rondom het testproces in de verschillende situaties. Omdat testen één van de instrumenten is om kwaliteit te waarborgen zal het testbeleid moeten aansluiten op de andere beleidsmaatregelen en initiatieven betreffende kwaliteitsmanagement. Het is aan te raden het testbeleid te laten aansluiten op het strategisch, tactisch en operationeel beleid van de organisatie.

Permanente testorganisatie

In een permanente testorganisatie wordt, in tegenstelling tot projectmatig testen, niet per project invulling gegeven aan een bepaald aspect van het testproces, maar over alle projecten heen. Redenen om een dergelijke organisatie in te richten zijn onder andere het beter kunnen benutten van schaarse expertise, het standaardiseren van testproducten, het beperken van de testproject opstarttijd, het continu willen verbeteren van het testproces, het consolideren van ervaringen en het vooraf inzicht willen hebben in testkosten en -doorlooptijd.

Testorganisatie in projecten

Bij aanvang van een testproject worden de rollen, taken, bevoegdheden en verantwoordelijkheden voor het testproject gedefinieerd. Dit kan voor het totale testproces (dus over alle testsoorten heen), of voor één bepaalde testsoort worden gedaan. Vervolgens wordt de samenhang tussen de rollen, de afzonderlijke testsoorten en de relaties met de andere betrokken partijen binnen het systeem ontwikkel proces bepaald en vastgelegd. De testmanager moet hierbij niet vergeten de relatie naar een eventuele test- of kwaliteitsafdeling te leggen.

De specifieke invulling van het bovenstaande is sterk gerelateerd aan de organisatievorm die voor het testen is gekozen.

De keuze is afhankelijk van de testsoort, project en organisatie. Soms, maar lang niet altijd, kan de testmanager hier invloed op uitoefenen. In grote lijnen zijn de volgende organisatievormen te onderkennen:

- testen als zelfstandige activiteit in project
- testen geïntegreerd in project
- testen als zelfstandige lijnorganisatie
- testen geïntegreerd in lijnorganisatie.

Testprofessional

Om als tester goed invulling te kunnen geven aan het testvak is een grote verscheidenheid aan expertise nodig. Zo moet een tester kennis hebben op het gebied van de materie, de infrastructuur (testomgeving, ontwikkelplatform, testtools) en het testen zelf. Wat is nu kenmerkend aan een tester of anders geformuleerd: welke eigenschappen moet een persoon hebben om deze de ideale tester te laten zijn? Hoewel er vele zijn te benoemen, moet de tester minimaal:

- communicatief, mondeling en schriftelijk vaardig zijn
- nauwkeurig kunnen werken en analytisch sterk zijn
- overtuigend zijn en een volhardende instelling hebben feitelijk zijn en een positief kritische houding hebben
- creatief zijn.

Testrollen

Het uitvoeren van testwerkzaamheden in een project of in de lijn vereist dat de taken gedefinieerd zijn en dat de uitvoerder van die taken beschikt over de juiste kennis en vaardigheden is.

Hierin zijn rollen en functies te onderscheiden. Een rol is de beschrijving van één of meer taken met de gewenste kennis en vaardigheden. Er zijn rollen die één op één overeenkomen met functies. Daarnaast zijn er rollen die niet voorkomen als functie.

Verschil en overeenkomst rol en functie:

- Een rol is gericht op het vervullen van taken voor het testproject of de permanente testorganisatie.
- Een functie is gericht op de medewerker en de plaats in de carrièrekubus.
- Gemeenschappelijk zijn de uit te voeren taken en de vereiste kennis en vaardigheden.

3.4 Adaptieve en complete methode

TMap is een aanpak die in alle testsituaties en in combinatie met elke willekeurige systeemontwikkelmethode toepasbaar is. Het biedt de tester een scala aan elementen voor zijn test, zoals: testontwerptechnieken, testinfrastructuur, teststrategie, fasering, testorganisatie, testtools, enzovoort. De tester kiest, afhankelijk van de situatie, de elementen uit TMap die hij gaat inzetten. Zo zijn er situaties waarin het voldoende zal zijn om slechts een beperkt aantal elementen in te zetten en zo zijn er situaties waar het juist noodzakelijk zal zijn alle elementen in te zetten. Dit maakt TMap als methode adaptief en dat wordt in deze context gedefinieerd als:

Definitie

Adaptief is het vermogen om een element op te splitsen in deelelementen die vervolgens anders gecombineerd opnieuw resulteren in een waardevol element voor de specifieke situatie.

Het adaptief zijn van TMap richt zich niet op een specifiek aspect van de methode, maar het zit door de hele methode verweven. Adaptief is meer dan het alleen kunnen inspelen op de veranderende omgeving. Het is ook het kunnen gebruiken van deze verandering ten voordele van het testen. Dit betekent dat TMap toepasbaar is in elke situatie én dat TMap toepasbaar is bij een veranderende situatie. Tijdens projecten en testen kunnen veranderingen optreden die impact hebben op eerder gemaakte afspraken. TMap biedt de elementen om met deze veranderingen om te gaan.

De adaptiviteit van TMap kan worden samengevat in vier adaptiviteitskenmerken:

- *Reageer* op veranderingen
- *(Her)gebruik* van producten en processen
- *Leer* van ervaringen
- *Probeer* voor gebruik

Deze kenmerken worden hierna toegelicht.

3.4.1 Reageer op veranderingen

Adaptiviteit begint met het vaststellen van de veranderingen en het reageren daarop. Binnen TMap vindt dit al meteen vanaf de start plaats bij de eerste activiteiten van het (master)testplan. Bij het vaststellen en oriënteren van de opdracht speelt het verkrijgen van inzicht in de omgeving waarin de test plaatsvindt en het vaststellen van eventuele mogelijke veranderingen een hoofdrol. Juist daar wordt de basis gelegd voor de verdere invulling en toepassing van de methode. Welke testsoorten, testvormen, fasen, tools worden op welke wijze ingevuld? Maar het blijft niet bij deze activiteiten. Het definiëren van de teststrategie met bijbehorende planning gebeurt in nauw overleg met de opdrachtgever. Zijn de opgestelde teststrategie en de hieruit opgestelde begroting en planning voor de opdrachtgever niet acceptabel, dan wordt het plan aangepast. De opdrachtgever heeft hierdoor nadrukkelijk grip op het testproces en kan deze sturen in de balans resultaat en risico versus tijd en kosten. Dit terugkoppelen vindt gedurende het gehele testtraject plaats en ook tijdens de beheerfase kan de testmanager in overleg met de opdrachtgever besluiten bepaalde zaken in het testplan aan te passen.

3.4.2 (Her)gebruik van producten en processen

Het snel kunnen gebruiken van producten en processen is voor adaptiviteit een vereiste. TMap biedt hier de mogelijkheid voor, door onder andere de grote hoeveelheid gereedschap die wordt meegeleverd in de vorm van testontwerptechnieken, checklists, sjablonen enzovoort. Deze zijn te vinden in het boek en op www.tmap.net.

Naast gebruik speelt hergebruik een belangrijke rol. Het zwaartepunt hiervan ligt in de fase Afronding waar de activiteiten zijn gedefinieerd om te identificeren wat er hergebruikt kan worden en hoe dit dan optimaal kan worden geconserveerd. Voor een organisatorische verankering van hergebruik van producten en processen reikt TMap diverse vormen van een permanente testorganisatie aan.

3.4.3 Leer van ervaringen

TMap biedt als methode de ruimte om te leren en het geleerde toe te passen.

Daarom is de activiteit evalueren testproces ingebed in het testproces. Een ander belangrijk instrument is het gebruik van metrics. Voor het testproces zijn metrics over de kwaliteit van het testobject en de voortgang van het testproces van groot belang. Ze worden gebruikt om het testproces te beheersen, om de testadviezen te onderbouwen en ook om systemen of testprocessen met elkaar te vergelijken. Voor het verbeteren van het testproces zijn metrics van belang om de gevolgen van bepaalde verbetermaatregelen te beoordelen.

3.4.4 Probeer voor gebruik

Binnen TMap is ruimte om uit te proberen voordat het werkelijk gebruikt gaat worden. De belangrijkste instrumenten hiervoor zijn de activiteiten rond de intake. De intake van de testbasis, de intake van de testinfrastructuur en de intake van het testobject bieden de ruimte om eerst uit te proberen en daarna pas werkelijk te gebruiken.

Het toepassen van TMap betekent niet dat direct alles uit het boek onverkort moet worden toegepast. Daarom betreft een andere vorm van proberen voor gebruik het ‘toesnijden’ van TMap op een bepaalde situatie. Hierbij kan een selectie worden gemaakt uit alle TMap elementen. Nadat de, op de eigen situatie toegesneden, aanpak is uitgeprobeerd (‘pilot’) kan deze in de organisatie worden uitgerold.

Voor veel situaties is dit ‘toesnijden’ van TMap al daadwerkelijk gebeurd. De specifieke TMap aanpak voor een bepaalde situatie (bekend onder de naam “toepassingsvariant”) kunt u vinden op www.tmap.net en in het TMap Test Topics boek

[Koomen, 2004].

Noten

1 BDTM is overigens niet een geheel zuivere benaming. Het woord “business” suggereert dat het alleen is bedoeld voor de link met de gebruikersafdelingen, terwijl testers natuurlijk nog steeds vaak genoeg enkel met de IT-afdelingen te maken hebben. In dit boek wordt echter de algemene benaming BDTM gebruikt.

2 Een productrisicoanalyse (PRA) heeft tot doel dat de verschillende belanghebbenden en de testmanager tot een gezamenlijk beeld komen van wat de meer en minder risicovolle delen/kenmerken van het systeem zijn. De focus bij de PRA ligt op de productrisico’s, oftewel: wat is het risico voor de organisatie wanneer het product niet de verwachte kwaliteit heeft.

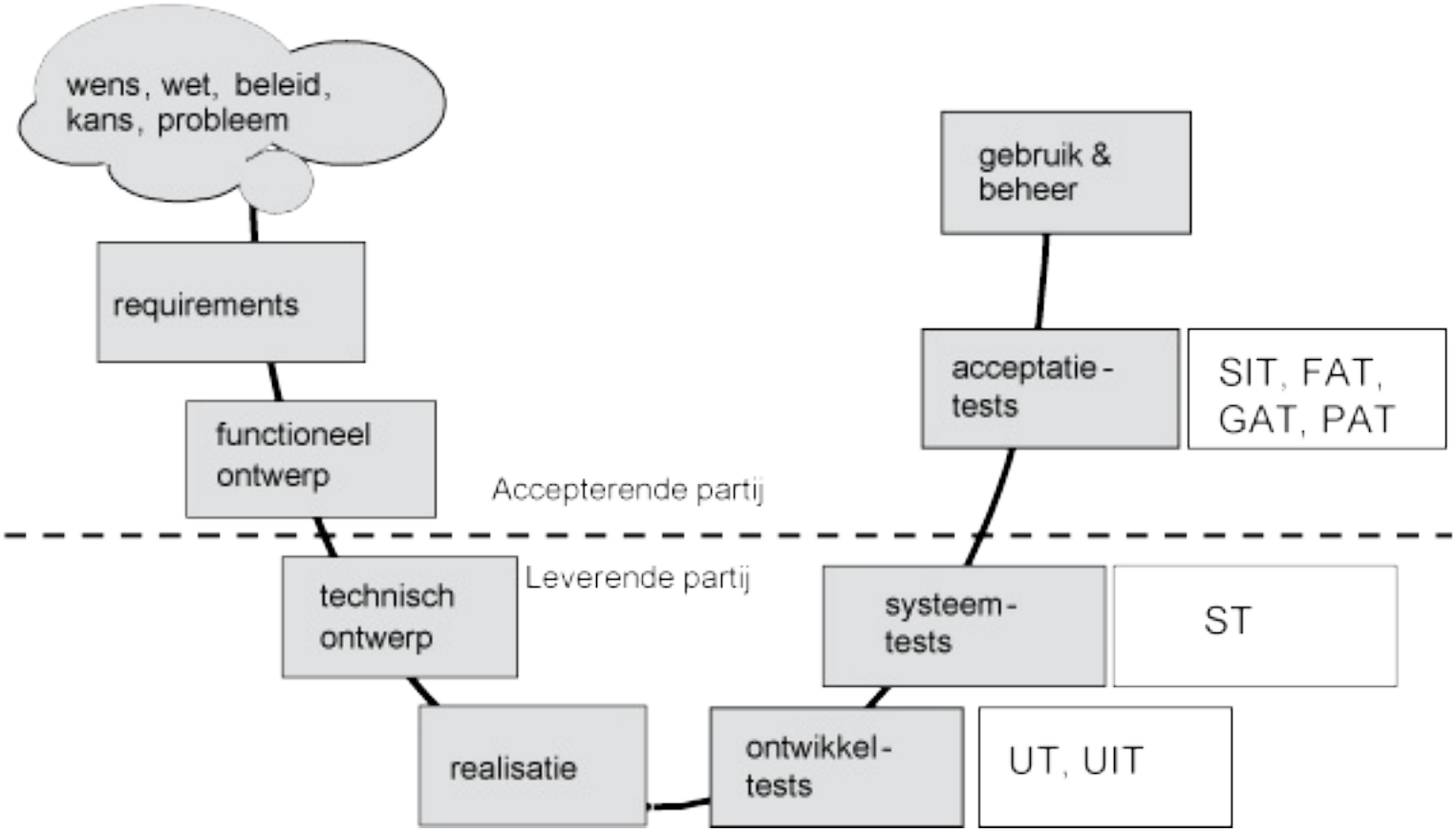
4 Inleiding processen

4.1 Indeling en opzet van de proceshoofdstukken

Dit hoofdstuk vormt een inleiding tot de hoofdstukken 5, 6, 7 en 8. Het testen is normaal gesproken georganiseerd in een aantal testsoorten. Iedere testsoort heeft een bepaald doel voor ogen, bijvoorbeeld het vaststellen of een component goed werkt of dat een systeem kwalitatief voldoende is om mee in productie te gaan. Op hoofdlijnen onderscheidt TMap de volgende groepen van testsoorten:

- ontwikkeltests
- systeemtests
- acceptatietests.

Deze groepen van testsoorten worden vaak ingevuld met meerdere afzonderlijke testsoorten, bijvoorbeeld de functionele, gebruikers- en productieacceptatietest zijn bekende acceptatietestsoorten. Hoewel in dit boek overwegend de begrippen ontwikkeltests, systeemtests en acceptatietests gebruikt worden, is in sommige gevallen meer detail nodig. In een dergelijke situatie wordt binnen TMap uit een vaste set aan testsoorten geput (figuur 4.1 “TMap testsoorten”). Dit is dus niet dé set, maar gewoon een set. Elke organisatie heeft vaak een eigen set aan testsoorten gedefinieerd. TMap hanteert binnen het boek vooral één bepaalde set van testsoorten om eventuele verwarring over testsoorten te voorkomen én om de testaanpak consistent te kunnen beschrijven.



Figuur 4.1: TMap testsoorten.

Onderstaand wordt een overzicht en een korte beschrijving van de TMap testsoorten gegeven. Hiermee kan een organisatie zelf een relatie te leggen met de testsoorten die binnen de organisatie worden gebruikt.

Definitie

- Unittest (UT)
  Een unittest is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een unit aan de in de technische specificaties gestelde eisen voldoet.
- Unitintegratietest (UIT)

Een unitintegratietest is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een logische groep units aan de in de technische specificaties gestelde eisen voldoet.

- Systeemtest (ST)
  Een systeemtest is een door de leverancier van de oplossing in een (goed beheersbare) laboratoriumomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem of delen daarvan aan de functionele- en niet-functionele specificaties en het technisch ontwerp voldoen.
- Systeemintegratietest (SIT)
  Een systeemintegratietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat (sub)systeeminterface afspraken zijn nagekomen, correct zijn geïnterpreteerd en correct zijn geïmplementeerd.
- Functionele acceptatietest (FAT)
  De functionele acceptatietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de functionele eisen voldoet.
- Gebruikersacceptatietest (GAT)
  De gebruikersacceptatietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de wensen/eisen van de gebruiker voldoet.
- Productieacceptatietest (PAT)
  De productieacceptatietest is een door de toekomstige beheerder(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de van uit beheer gestelde eisen voldoet.

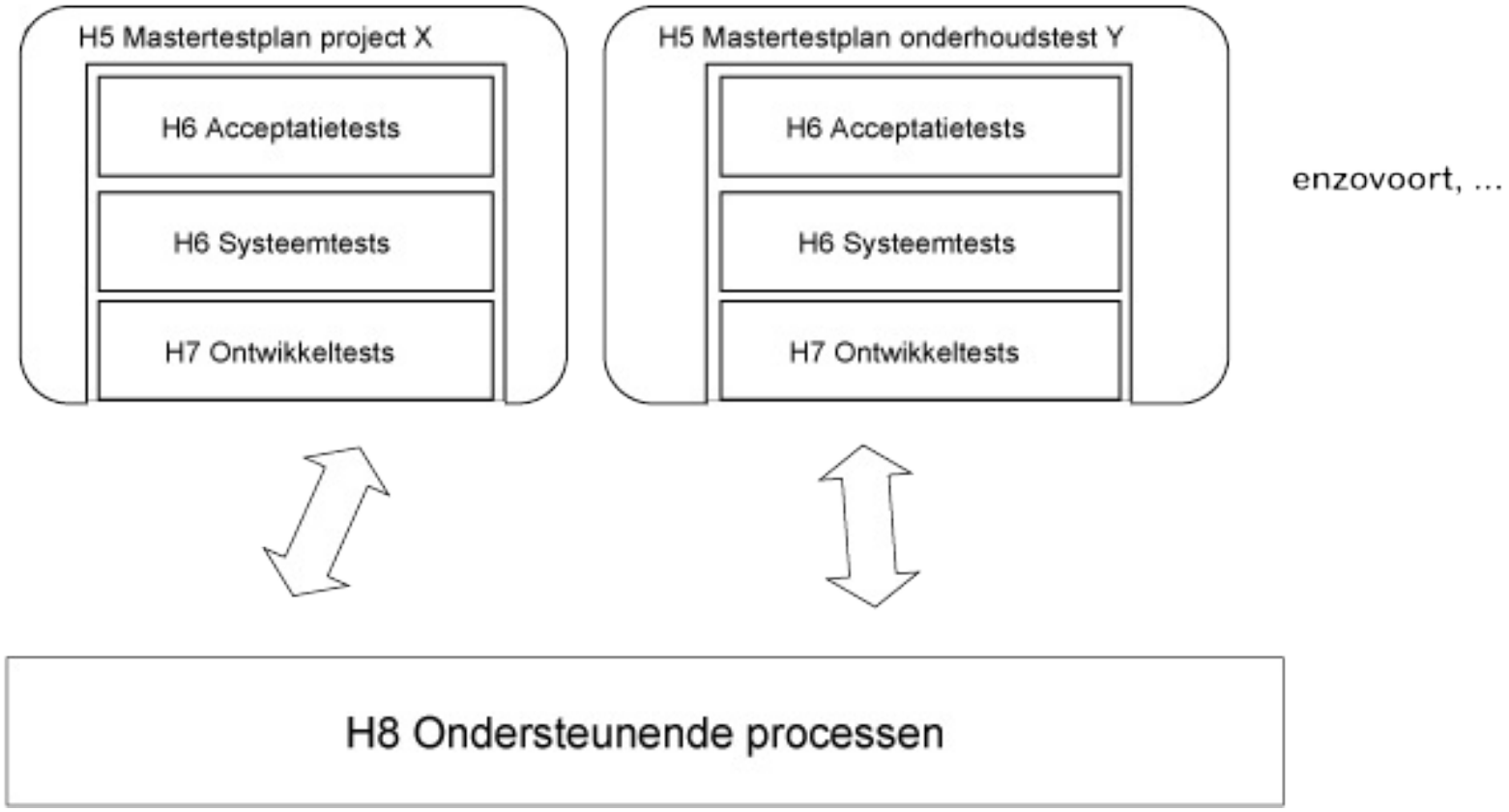
De testsoorten dienen onderling afgestemd en gecoördineerd te worden. Dit gebeurt door het opstellen van een mastertestplan en het beheren van het overkoepelende testproces.

Zowel voor het mastertestplan als de testsoorten geldt dat het belangrijk is om voor het maken van plannen, het voorbereiden, uitvoeren en beheren van activiteiten een goed proces in te richten. In beide gevallen zijn deze processen gericht op het testen van een bepaald softwareproduct (administratief systeem of embedded software, zelf gebouwd of pakket). Deze processen kunnen toegepast worden in een testproject of in een test vanuit een lijnafdeling, bijvoorbeeld een onderhoudstest van een periodieke nieuwe release.

Daarnaast hebben organisaties vaak lijnondersteuning voor het testen ingericht, die losstaat van een specifieke test of specifiek testobject. Vormen van algemene ondersteuning zijn bijvoorbeeld een afdeling met testadviseurs die de individuele testprojecten ondersteunen, een resourcepool met testers waar de projecten uit kunnen putten tot zelfs complete testfabrieken. Specifiekere vormen van ondersteuning vanuit de lijn zijn het selecteren en beheren van testtools en het inrichten en beheren van testomgevingen. Ook deze ondersteunende activiteiten kunnen als proces gezien en ingericht worden.

In de hierop volgende hoofdstukken zijn de volgende processen beschreven (figuur 4.2, “TMap processen”):

- Mastertestplan, managen van het totale testproces (hoofdstuk 5);
- Acceptatie- en systeemtesten (hoofdstuk 6);
- Ontwikkeltesten (hoofdstuk 7);
- Ondersteunende processen (hoofdstuk 8).



Figuur 4.2: TMap processen.

De volgorde van de hoofdstukken is een tijdsvolgorde: als eerste wordt het mastertestplan opgesteld, daarna de testplannen voor acceptatie- en systeem tests en tijdens of kort voor realisatie de plannen voor de ontwikkeltests. De ondersteunende processen hebben niet een echt begin- of eindpunt maar moeten meer als continue processen gezien worden.

Wellicht opvallend is dat de testsoorten acceptatie- en systeemtest niet apart maar als één proces worden beschreven. Hier is voor gekozen omdat de activiteiten in beide testsoorten vrijwel hetzelfde zijn en aparte procesbeschrijvingen daardoor (te)veel overlap zouden vertonen. Zowel de acceptatietest als de systeemtest zijn feitelijk te beschouwen (en daarmee ook te organiseren) als op zich staande processen. Ze hebben een eigen testplan, een eigen budget en vaak ook een eigen testomgeving waarop de test wordt uitgevoerd. Het zijn processen die parallel lopen aan het ontwikkelproces en die dienen te starten tijdens het opstellen van de functionele specificaties.

De activiteiten voor het opstellen en beheren van een mastertestplan hebben ook veel overeenkomsten met het opstellen en beheren van een acceptatie- of systeemtestplan. Met name door het hogere beschouwingniveau, waarbij het mastertestplan als het ware de andere testplannen omvat, wijken de activiteiten toch ook op allerlei punten af. Om die reden verdient het een eigen hoofdstuk. Dit hoofdstuk is zo opgezet dat de volledige activiteiten zijn opgenomen, maar de beschrijvende en toelichtende teksten met name gericht zijn op de specifieke aspecten van mastertestplanning en beheersing van het overkoepelende testproces. De basisuitleg en veel tips zijn opgenomen bij de beschrijvingen van acceptatie- en systeemtesten in paragrafen 6.2 “Fase Planning” en 6.3 “Fase Beheer”. Hier is voor gekozen om een aantal redenen:

- om de tijdsvolgorde in de hoofdstukken te handhaven;
- de verwachting is dat hoofdstuk 6 vaker en grondiger bestudeerd en gebruikt gaat worden dan hoofdstuk 5;
- de testmanager van een mastertestplan heeft normaal gesproken al het carrièrepad van testmanager van een acceptatie- of systeemtest doorlopen en is hier bekend mee.

De lezer wordt daarom aangeraden eerst paragrafen 6.2 en 6.3 te lezen alvorens aan hoofdstuk 5 te beginnen.

4.2 Hoofdstukken 5, 6 en 7: de TMap fasen

De processen en onderliggende activiteiten in hoofdstukken 5, 6 en 7 worden beschreven aan de hand van de TMap fasen. Voor hoofdstuk 5 zijn dit de fasen Planning en Beheer. Hoofdstuk 6 en 7 worden besproken aan de hand van het complete TMap-faseringsmodel. Dit is een generiek model dat de verschillende activiteiten overzichtelijk in kaart brengt, met hun onderlinge volgorde en afhankelijkheden. In het TMap faseringsmodel zijn de testactiviteiten verdeeld over een

zevental fasen. Dit zijn de fasen Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding.

In hoofdstuk 5 en 6 worden de fasen, processen en activiteiten in detail beschreven. De beschrijving van de fasering voor de procesbeschrijvingen van achtereenvolgens mastertestplan (managen van het totale testproces), acceptatie- en systeemtesten is als volgt:

Per *fase* is aangegeven wat het doel van de fase is, wat de context is, onder welke randvoorwaarden de fase wordt uitgevoerd, welke rollen en verantwoordelijkheden van belang zijn en uit welke activiteiten de fase is opgebouwd. Daarnaast is een korte beschrijving van de werkwijze opgenomen.

Per *activiteit* is aangegeven wat het doel is, welke werkwijze wordt gehanteerd en welke producten worden opgeleverd. Voor zover van toepassing is eveneens de relatie gelegd met testtechnieken en -tools.

De uitklapbare flap bevat een overzicht van de processen, fasen en activiteiten van de hoofdstukken 5 en 6.

Hoofdstuk 7 bevat een beknopte beschrijving van de voor ontwikkeltesten benodigde fasen en activiteiten. Dit hoofdstuk is meer beschouwend van aard en maakt ook een vergelijking met de overige testsoorten. Het is minder een “hoe”-hoofdstuk dan 5 en 6. De doelgroepen zijn met name:

- de ontwikkelaars/ontwikkeltesters die op zoek zijn naar ideeën;
- de testadviseur die gevraagd wordt om (de inrichting van) het ontwikkeltesten te ondersteunen;
- de testmanager voor het overkoepelende testproces die de ontwikkeltests moet afstemmen met de andere testsoorten;
- de lijn- of projectmanager van de ontwikkelaars, die geïnteresseerd is in het beter beheersen van de kwaliteit van de geproduceerde software en wil weten hoe dit bereikt kan worden.

4.3 Hoofdstuk 8, ondersteunende processen

In de hoofdstukken 5, 6 en 7 wordt gesproken over de testprocessen die werken volgens het faseringsmodel van TMap en hoe dit georganiseerd wordt voor het testen van een systeem of pakket. Het laatste proceshoofdstuk gaat in op ondersteunende processen op organisatieniveau. Dit valt uiteen in een aantal afzonderlijke onderwerpen:

- Testbeleid
  Een organisatie formuleert een testbeleid hoe omgegaan wordt met de mensen, middelen en methoden rondom de testprocessen. Dit bepaalt tevens welke centrale ondersteuning ingericht wordt;
- Permanente testorganisatie
  Er zijn diverse vormen van een permanente testorganisatie, zoals een adviesgroep, een resourcepool met testers of een zogenaamde testfabriek. Het verschijnsel wordt uitgebreid toegelicht, inclusief de mogelijke diensten, er wordt een algemeen procesmodel van een dergelijke organisatie gegeven en het proces van het opzetten van een testorganisatie wordt kort toegelicht;
- Testomgevingen
  Het inrichten en beheren van de testomgeving is vaak belegd bij een aparte afdeling, buiten het project. Diverse aspecten van testomgevingen worden belicht, inclusief een aantal processen die goed ingericht moeten zijn om een beheersbare testomgeving te kunnen garanderen. Ook wordt het inrichten en beheren van testomgevingen als dienst besproken;
- Testtools
  Testtools worden vaak op organisatieniveau aangeschaft en beheerd. De individuele testprojecten kunnen/moeten dan van deze tools gebruik maken. Het fenomeen testools wordt in algemene zin besproken, inclusief het proces om een testtool in te voeren.
- Testprofessionals
  In veel organisaties worden de testprojecten bemenst vanuit een afdeling of pool met testers of worden externe testers ingehuurd. Aan een tester moeten eisen worden gesteld. Het bieden van een carrièrepad inclusief opleidingen moet in het reguliere HRM-proces worden ingebed om de testers te kunnen werven én houden voor de organisatie.

5 Mastertestplan, managen van het totale testproces

5.1 Inleiding

In paragraaf 2.3 “Plaats van het testen” is al ingegaan op het feit dat het testen van software (informatiesysteem, pakketimplementatie of embedded software) normaal gesproken wordt georganiseerd met een aantal testsoorten. Iedere testsoort heeft een bepaald doel voor ogen, bijvoorbeeld het vaststellen of een component goed werkt of dat een systeem kwalitatief voldoende is om mee in productie te gaan.

Wanneer de testmanager van iedere testsoort met zijn/haar directe afnemers bepaalt wat getest gaat worden, is de kans groot dat in het totaalbeeld van het testen bepaalde zaken onnodig dubbel getest worden óf dat juist bepaalde aspecten tussen wal en schip vallen. De werkwijze moet andersom zijn: vanuit het totaaloverzicht maakt een testmanager in overleg met opdrachtgever en andere betrokkenen een verdeling welke testsoort wat wanneer test (en hoe zwaar), passend binnen de gehanteerde ontwikkel- of onderhoudsaanpak. Het streven is hierbij om de belangrijkste fouten zo vroeg en goedkoop mogelijk te ontdekken. Deze afstemming wordt vastgelegd in het zogenaamde mastertestplan (MTP). Dit plan vormt de basis voor de detailtestplannen van de afzonderlijke testsoorten.

Naast deze inhoudelijke afstemming zijn andere redenen om af te stemmen het hanteren van uniformiteit in processen (bijvoorbeeld de bevindingenprocedure en het beheer van de testware), beschikbaarheid en beheer van de testomgeving en tools, en optimale verdeling van de resources (zowel mensen als middelen) over de testsoorten heen.

Uitgediept

Mastertestplan vanzelfsprekend?

Het opstellen van een mastertestplan klinkt misschien vanzelfsprekend, maar is het in de praktijk nog niet. De afstemming wordt helaas vaak overgelaten aan de testmanagers van de individuele testsoorten. Afstemming lukt dan meestal nog wel op het niveau van mijlpalen, maar wordt moeilijker wanneer het gaat om de scope en zwaarte van de verschillende testsoorten. Een projectmanager kan hier doorgaans onvoldoende aandacht aan geven, de testmanager van een afzonderlijke testsoort heeft hiervoor onvoldoende mandaat. Gelukkig wordt ook binnen de systeemontwikkeling steeds vaker het belang ingezien van het op elkaar afgestemd krijgen én houden van de testsoorten. Zo is binnen IBM’s systeemontwikkelmethode Rational Unified Process® het mastertestplan een officieel product.

Opstellen van een mastertestplan en beheren van het totale testproces vraagt om een speciale rol: de testmanager of overall testcoördinator voor het overkoepelende testproces.

“Onnodig dubbel getest”

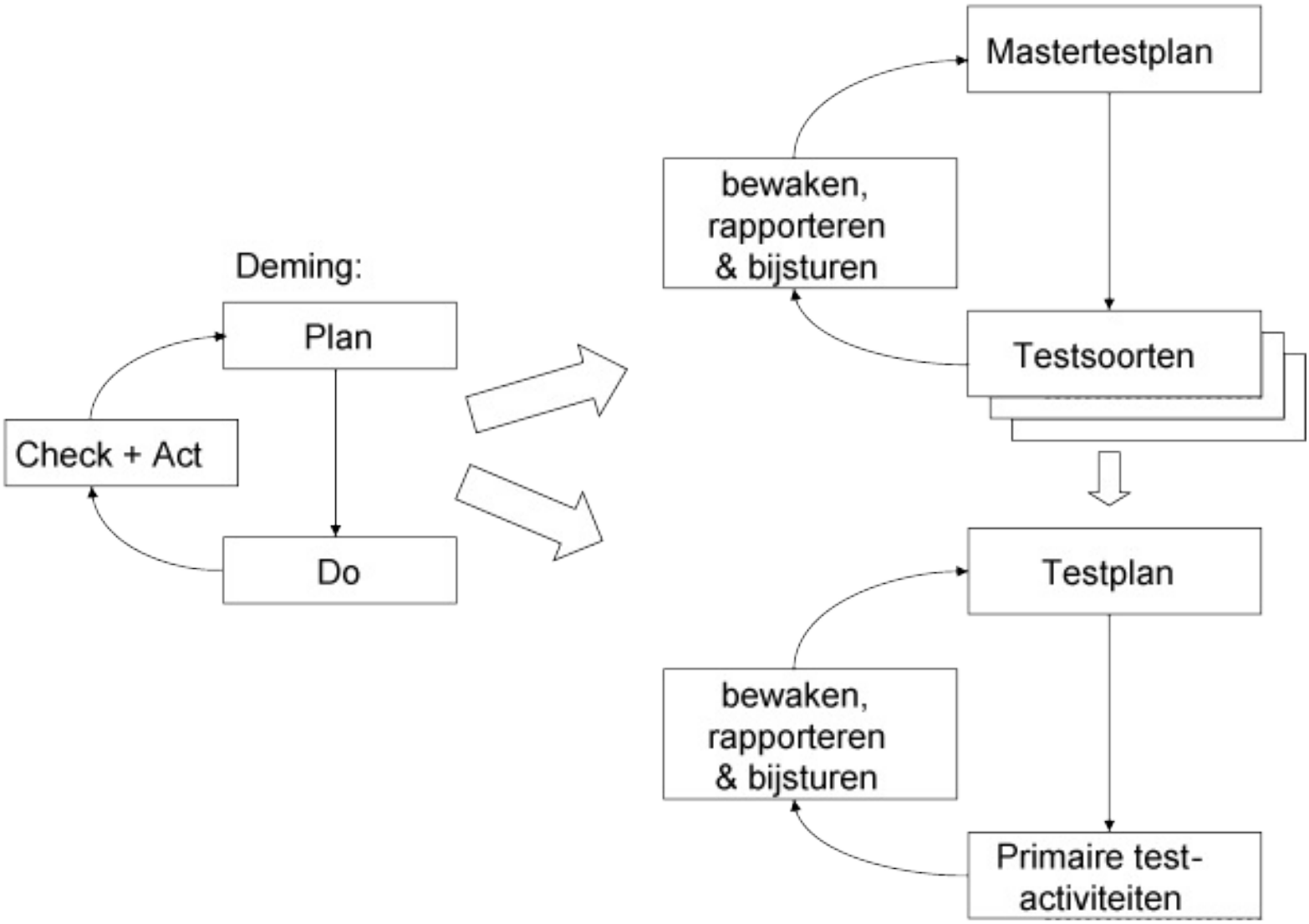
Als belangrijke reden voor een mastertestplan wordt genoemd het vermijden van onnodig dubbel testen. Het woord “onnodig” is hierbij belangrijk. Dubbele tests zijn op zich namelijk niet erg, vaak ook niet te vermijden en in sommige gevallen zelfs verplicht. Zo zal in een unittest vaak dezelfde functionaliteit geraakt worden als in een systeemtest en zullen diverse testgevallen op elkaar lijken. Ook kan een deel van de systeemtest bewust herhaald worden in de acceptatietest, bijvoorbeeld om te controleren of alles werkt op de productie-like infrastructuur of binnen de bestaande bedrijfsprocessen. Een voorbeeld van “onnodig dubbel” is wanneer twee testsoorten een gelijke testontwerptechniek gebruiken om gelijke testgevallen af te leiden en uit te voeren.

Dit hoofdstuk gaat over het beheersen van het totale testproces vanuit de BDTM-filosofie en beperkt zich dus niet tot het maken van het mastertestplan. Juist bij het uitvoeren van de testplannen is coördinatie en bijsturing essentieel. Verlate opleveringen, tegenvallende kwaliteit of gebrek aan tijd of resources zijn eerder regel dan uitzondering in de IT. De testmanager van een testsoort moet dan de geplande aanpak bijsturen. Het over de testsoorten heen coördineren en sturen van het totale testproces moet ervoor zorgen dat ondanks individuele bijsturingen in verschillende testsoorten er nog steeds een coherente totaalaanpak van het testen blijft bestaan. Bijsturing in een testsoort heeft mogelijk inefficiëntie tot gevolg bij andere testsoorten, bijvoorbeeld omdat het testobject met onvoldoende kwaliteit de eerste testsoort verlaat. De testmanager moet dit vervolgens signaleren en maatregelen voorstellen of nemen, in overleg met opdrachtgever en projectmanagement. In het sturings- en verbetermechanisme dat hiervoor gehanteerd wordt is het zogenaamde Deming-

wiel [Deming, 1992] te herkennen: Plan iets (Plan), Doe het (Do), Controleer het resultaat (Check) en Stuur zo nodig bij (Act). Zie figuur 5.1.

Het hoofdstuk wordt gesplitst in de fase planning (met behulp van een mastertestplan) en in de fase beheer. Voor elk onderwerp wordt eerst een overzicht gegeven, waarna de activiteiten in meer detail worden uitgewerkt. Bij dit hoofdstuk wordt de lezer aangeraden eerst paragrafen 6.2 en 6.3, het plannen en beheren van systeem- en acceptatietesten, te lezen. Veel van de zaken en activiteiten die daarin beschreven staan, gelden ook voor dit hoofdstuk.

In sommige gevallen, met name bij onderhoud en outsourcing, worden grote delen van het mastertestplan herbruikbaar opgezet. Dit is als laatste in dit hoofdstuk beschreven als variant, de Generieke Test Afspraken (GTA).



Figuur 5.1: Sturing mastertestplan en testsoorten.

5.2 Fase Planning van het totale testproces

Doel

Het inrichten van het totale testproces door:

- de testsoorten op elkaar af te stemmen;
- overlap of gaten in de testdekking te minimaliseren;
- beschikbare middelen optimaal te verdelen, zoals:
  - testers;
  - infrastructuur en tools;
  - technische- of materiedeskundigheid;
- de belangrijkste fouten zo vroeg mogelijk te vinden;
- het testen zo kort mogelijk op het kritieke pad van het totale project te laten plaatsvinden;
- uniformiteit in de testprocessen te bewerkstelligen;

- afspraken met betrokkenen vast te leggen;
- de opdrachtgever te informeren over de aanpak, planning, begroting, activiteiten en de op te leveren (eind)producten met betrekking tot het totale testproces.

Het plan geeft inzicht in de diverse toe te passen test- en toetssoorten zodanig dat er een optimalisatie plaatsvindt van het totale testproces. Alle andere testplannen dienen afgeleid te worden van dit mastertestplan. Het mastertestplan is daarmee sturend voor de onderliggende testsoorten.

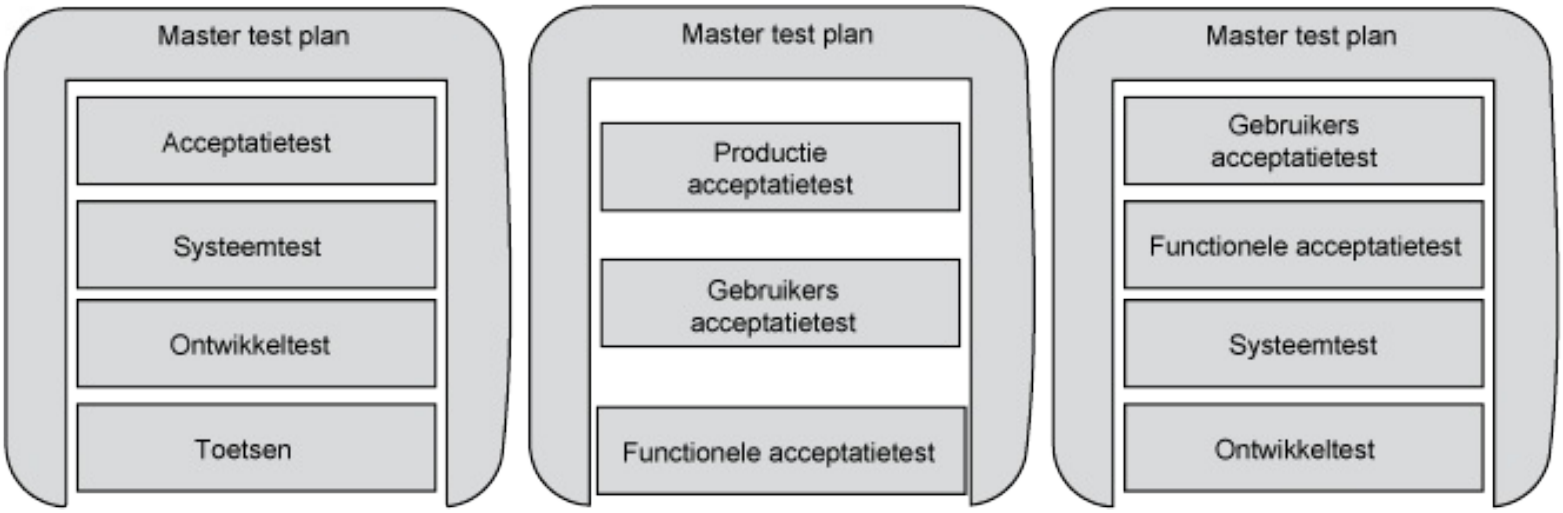
Context

Een mastertestplan is nodig wanneer er meerdere testsoorten zijn. In de praktijk is dit meestal het geval, zowel bij nieuwbouw als bij onderhoud of bij een pakketimplementatie, hetzij waterval, hetzij iteratief.

Uitgediept

Het soort systeem en/of de ontwikkelaanpak en/of het opgestelde testbeleid bepalen welke testsoorten in welke vorm het beste toegepast kunnen worden. Zo is bij iteratieve systeemontwikkeling een zware acceptatietest minder voor de hand liggend. Dit komt doordat de kwaliteit vanuit gebruikersoogpunt al in voorgaande testsoorten is getest. Bij pakketimplementaties ligt juist weer veel meer nadruk op een grondige acceptatietest. De risico's zitten daar met name in de implementatie van het pakket in de organisatie, een typisch acceptatietest aspect.

Na het vaststellen welke testsoorten er zijn of moeten komen, moet vervolgens gekeken worden welke van deze testsoorten betrokken worden in het mastertestplan. In theorie komen alle testsoorten en ook toetssoorten (reviews, inspecties) in aanmerking, maar in de praktijk zijn het vaak de testsoorten als systeem- en acceptatietest (zie onderstaande figuur) die in het mastertestplan op elkaar afgestemd worden. Zie figuur 5.2 voor enkele voorbeelden van welke testsoorten onder een mastertestplan kunnen vallen.



Figuur 5.2: Voorbeelden van beschouwingsgebieden van een mastertestplan.

Om test- en toetssoorten in een mastertestplan op elkaar af te stemmen, moet elke test- of toetssoort inzicht geven in wat ze wel of juist niet test en met welke diepgang. De afstemming wordt echter moeilijk tot onmogelijk wanneer dit inzicht niet gegeven wordt en van bijvoorbeeld de ontwikkeltest alleen bekend is dát een programmeur test, maar niet hóe. Kan of wil men dit inzicht voor een bepaalde testsoort niet leveren, dan wordt het erg moeilijk deze testsoort te betrekken bij het mastertestplan. Hier ligt dan een taak voor project- en testmanagement om dit te veranderen. Dit geldt zeker als het mastertestplan bedoeld is om voor te schrijven wat en hoe de testsoorten moeten testen.

Uitgediept

Ontwikkeltests en toetssoorten

Het is overigens wél belangrijk om de ontwikkeltests en de toetssoorten te betrekken in het mastertestplan, omdat dit allerlei mogelijkheden tot optimalisatie van het totale proces biedt. Zowel de ontwikkeltests als toetsen vinden eerder in de systeemontwikkeling plaats dan de systeem- en acceptatietests en zijn als zodanig in staat fouten eerder

en dichter bij de bron te detecteren, bij toetsen zelfs voordat de fouten in de software geïmplementeerd zijn. Dit betekent dat de herstelkosten lager zijn en er eerder begonnen kan worden met het adviseren over de kwaliteit van het systeem. De toetssoorten dienen dus met grote voorkeur onderdeel van het plan te zijn. Om te vermijden dat in dit hoofdstuk steeds over "testen en toetsen" wordt gesproken, wordt enkel de term "testen" gehanteerd. Hier wordt dan uitdrukkelijk ook toetsen bedoeld.

Afstemming ongewenst?

Het is mogelijk dat de verantwoordelijke partij voor een bepaalde testsoort de afstemming (terecht of onterecht) ervaart als ongewenste invloed van andere partijen op de eigen test. Het is echter nog steeds zinvol te proberen deze testsoort bij het mastertestplan te betrekken, omdat het mastertestplan bijvoorbeeld een dubbele test op hetzelfde aspect of juist een ontbrekende test van een aspect expliciet zichtbaar maakt. Als bijvoorbeeld een derde partij het systeem ontwikkelt en systeemtest, zoals bij outsourcing vaak gebeurt, zijn er soms onvoldoende mogelijkheden de testsoorten vooraf op elkaar af te stemmen. In het mastertestplan moeten dan wel de eisen worden opgenomen die gesteld worden aan de testsoorten bij de leverancier. Naast eisen over de te volgen teststrategie omvatten deze bijvoorbeeld ook oplevering van testware, testresultaten en rapportages. Het mastertestplan dient dan tevens als communicatiemiddel naar deze partij.

Verder moet vroeg in het systeemontwikkeltraject met het mastertestplan begonnen worden, zodat zoveel mogelijk testsoorten betrokken kunnen worden. "Vroeg" in dit verband kan zijn tijdens het opstellen van de requirements of zelfs de initiatie van het project. Hoe later in het traject het mastertestplan start, hoe minder mogelijkheden er zijn om de testsoorten optimaal in te richten en op elkaar af te stemmen.

Het opstellen en uitvoeren van een mastertestplan kan plaatsvinden ongeacht de "testvolwassenheid" van de te betrekken testsoorten, ofwel de kwaliteit van hun testprocessen. Wel geldt dat als de testvolwassenheid laag is, de testmanager er rekening mee moet houden dat hij in het mastertestplan niet te hoge eisen kan stellen aan inrichting en beheer van de afzonderlijke testsoorten.

Een mastertestplan wordt opgesteld in de bredere context van een proces voor systeemontwikkeling of -onderhoud of pakketimplementatie. Daarbinnen worden ook diverse andere plannen opgesteld, bijvoorbeeld een Project Initiatie Document (bij de PRINCE2® projectmanagementmethode [PRINCE2, 2002]), het project of werkplan, het configuratiemanagementplan en een kwaliteitsplan. De testmanager moet uitzoeken welke andere plannen er zijn en welke relaties er zijn tussen het mastertestplan en deze plannen.
Voorbeelden:

- De begroting in het mastertestplan is input voor het projectvoorstel waarin om budget wordt gevraagd.
- In het configuratiemanagementplan zijn ook de testproducten opgenomen. De testmanager moet hierop invloed uitoefenen om te zorgen dat de testproducten beheerd kunnen worden zoals hij dat wil.
- De detailintake van de testbasis kan plaatsvinden in de reguliere reviews. In het kwaliteitsplan moet dan deelname van de testers aan de reviews geregeld zijn.

Randvoorwaarden

Om te kunnen starten met het opstellen van het mastertestplan dienen de volgende zaken bekend te zijn:

- Doel en belang van het systeem of pakket voor de organisatie
- Globale eisen / requirements
- De organisatie van het ontwikkelproces
- Globale (oplever)planning
- Globale systeemomvang (in functiepunten of uren)
- De werkwijze voor het te ontwikkelen of te onderhouden systeem of te implementeren pakket
- De opdrachtgever voor het totale testproces.

Wanneer deze informatie nog niet beschikbaar is, bijvoorbeeld omdat de ontwikkelaanpak nog onbekend is, heeft dit negatieve gevolgen. Ofwel de doorlooptijd of benodigde inspanning voor het opstellen van het plan worden langer respectievelijk meer, ofwel de kwaliteit en gewenste mate van detail van het plan worden minder.

Ook moet er de bereidheid en mogelijkheid zijn tot het maken van algemene afspraken op gebied van testen.

Werkwijze

De testmanager als opsteller van het mastertestplan ondersteunt de opdrachtgever bij het opstellen van een heldere opdracht, rekening houdend met de vier BDTM-aspecten Resultaat, Risico's, Tijd en Kosten (zie ook paragraaf 3.1 "Business driven toegelicht"), en legt deze vast in het plan. Vervolgens gaat de testmanager zich inwerken in het aankomende traject door gesprekken met belanghebbenden te voeren en informatiebronnen zoals documentatie te raadplegen. Parallel hieraan vult de testmanager de opdracht verder in en bepaalt samen met de opdrachtgever het beschouwingsgebied. Er wordt samen met belanghebbenden een productrisicoanalyse uitgevoerd om vast te stellen wat de gewenste resultaten van het testen moeten zijn en welke delen en aspecten van het te testen systeem of pakket meer of minder risicovol zijn.

Deze analyse vormt de basis voor de teststrategie.

Nu ontstaat een traject met iteratieve kenmerken. In de teststrategie wordt bepaald welke testsoorten ingericht moeten gaan worden, krijgen de verschillende testsoorten richting wat er getest moet worden en met welke diepgang (hoe meer risico, hoe zwaarder de test). Vervolgens worden de testsoorten en het overkoepelende testproces op hoofdlijnen begroot en in een planning uitgezet (hoe meer risico, hoe eerder). Dit wordt afgestemd met de opdrachtgever en andere betrokkenen en afhankelijk van hun oordeel mogelijk herzien. In dat geval worden de stappen opnieuw doorlopen. Conform BDTM heeft de opdrachtgever dus nadrukkelijk grip op het testproces.

Verdere stappen in de planvorming zijn dat de testmanager de producten definieert die vanuit de testsoorten opgeleverd moeten worden en een voorstel doet hoe de testorganisatie eruit moet gaan zien, op centraal niveau en globaal per testsoort. Hij stemt de infrastructuurbehoeften van de testsoorten onderling af en definieert de benodigde testinfrastructuur. Testbeheer kan deels al op mastertestplan-niveau ingericht worden. Dit kan zowel door centraal procedures en standaarden voor beheer op te stellen als door het centraal beheren van bijvoorbeeld bevindingen en testproducten. Beide mogelijkheden hebben tot doel te voorkomen dat het wiel in de afzonderlijke testsoorten opnieuw uitgevonden wordt. De belangrijkste (project)risico's die het testproces bedreigen worden benoemd en er worden mogelijke maatregelen voorgesteld om deze risico's te beheersen. Als laatste stap laat de testmanager het mastertestplan goedkeuren door de opdrachtgever en eventuele andere belanghebbenden.

Hoewel de activiteiten in dit deelproces opvolgend beschreven zijn, zullen in de praktijk bepaalde activiteiten meerdere malen en/of in andere volgorde doorlopen worden. Wanneer bijvoorbeeld de voor een test benodigde infrastructuur niet geleverd kan worden of te duur wordt bevonden, moet misschien de teststrategie aangepast worden.

Tip

(Te) vroege betrokkenheid?

Het streven is dat de testmanager zo vroeg mogelijk in het ontwikkeltraject betrokken wordt om het mastertestplan op te stellen. Dit geeft de meeste mogelijkheden om het totale testen goed vorm te geven. Een nadeel van deze vroege betrokkenheid is dat de testmanager soms zo vroeg betrokken is dat er na het mastertestplan even niet zoveel meer te doen lijkt. Ontwerpen zijn nog niet opgeleverd, waardoor het nog te vroeg is om testgevallen te gaan specificeren. In de praktijk zijn er, behalve part-time inzet, een aantal mogelijke taken die de testmanager in overleg met opdrachtgever binnen het traject op zich kan nemen:

- Opstellen van testplannen van afzonderlijke testsoorten
- Opstellen van procedures, maken van sjablonen, opzetten bevindingenadministratie
  De in het mastertestplan beschreven beheeraanpak heeft vaak verdieping nodig.
  De testmanager kan dit goed invullen.
- Organiseren van reviews/inspecties
  Deze activiteit is vaak niet belegd of is belegd bij de projectmanager die er (te) weinig tijd voor heeft. De testmanager is in een goede positie om deze activiteit te organiseren, sessies te modereren, enzovoort.
- Reviewen van tussenproducten
  Door zelf deel te nemen aan reviews van tussenproducten vanuit oogpunt van testbaarheid wordt de kwaliteit van de tussenproducten verhoogd en doet de testmanager veel kennis op over het te testen systeem. Speciaal het vermelden waard is het reviewen of relevante niet-functionele requirements zoals beveiliging en gebruikersvriendelijkheid wel gespecificeerd zijn en of de requirements voldoende SMART (Specifiek, Meetbaar, Acceptabel, Realistisch en Tijdgeboden) gedefinieerd zijn.
- Taken binnen requirements-, kwaliteits-, configuratie-, change of risicomanagement

Indien de testmanager hiervan kennis heeft, kan deze mogelijk een rol vervullen bij het opzetten of uitvoeren van andere aan kwaliteitsborging gerelateerde projectactiviteiten.

- Verhoog bewustwording testen
  Besteed tijd aan de bewustwording in de organisatie en het project van (het belang van) testen. Maak de plaats van testen én de rol van de testmanager in het komende traject duidelijk. Het aanleggen van een sociaal netwerk en het nadrukkelijk opeisen van de testmanager-rol versterkt de positie van de testmanager in het verdere traject.

Rollen/verantwoordelijkheden

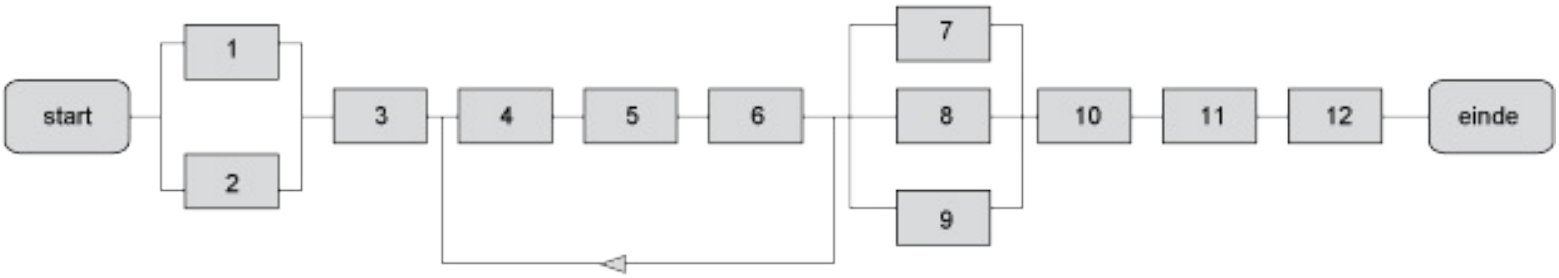
De primair verantwoordelijke rol voor het opstellen van het mastertestplan is de testmanager, soms ook overall testcoördinator genaamd. In bepaalde gevallen is de rol gecombineerd met quality assurance (QA-)verantwoordelijkheid tot integrale test-en QA-manager.

Activiteiten

Het opstellen van het mastertestplan omvat de volgende activiteiten:

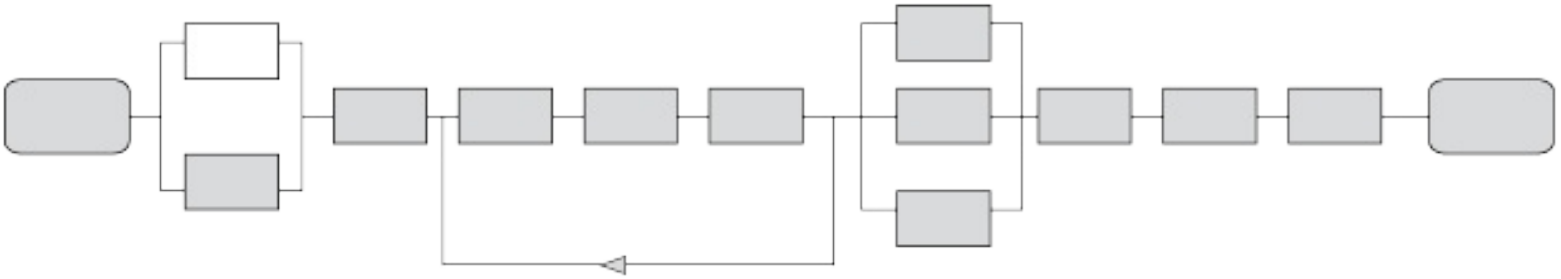
1. Vaststellen opdracht;
2. Oriënteren opdracht;
3. Analyseren productrisico's;
4. Bepalen teststrategie;
5. Bepalen begroting;
6. Bepalen planning;
7. Definiëren testproducten;
8. Definiëren organisatie;
9. Definiëren infrastructuur;
10. Inrichten beheer;
11. Bepalen testprocesrisico's en maatregelen;
12. Terugkoppelen en fixeren plan.

Onderstaand schema geeft de globale volgorde en de afhankelijkheden aan tussen de verschillende activiteiten. Voor alle activiteiten geldt dat deze meerdere malen doorlopen kunnen worden omdat de resultaten van een activiteit herziening van een voorgaande activiteit kunnen betekenen. Zoals al in de werkwijze is aangegeven, hebben de stappen 4, 5 en 6 een expliciet iteratief karakter:



Figuur 5.3: Opstellen mastertestplan.

5.2.1 Vaststellen opdracht



Doel

Een testproces begint met een formulering van de opdracht zodat het doel, de taken, verantwoordelijkheden en bevoegdheden van het testen voor alle betrokkenen duidelijk worden.

Werkwijze

De opdrachtformulering is één van de essentiële onderdelen van een testproces. Met het vastleggen van de opdrachtformulering in het mastertestplan worden afspraken over het totale testproces met de belanghebbenden (onder andere de opdrachtgever) expliciet gemaakt. Verwachtingen worden over en weer op elkaar afgestemd.

De opdrachtformulering in het mastertestplan vormt de overkoepelende opdracht voor alle onderliggende testsoorten. De opdrachtformulering van elke testsoort moet hierop aansluiten.

Een opdrachtformulering voor een mastertestplan bestaat uit de volgende onderdelen:

- Opdrachtgever
- Opdrachtnemer
- Opdracht
- Beschouwingsgebied
- Randvoorwaarden en uitgangspunten.

Deze onderdelen worden hieronder nader toegelicht:

Opdrachtgever

De partij die de opdracht geeft tot het opstellen van het mastertestplan en het uitvoeren van de tests. Het is belangrijk voor het testproject om te onderkennen wie de opdracht geeft voor het uitvoeren van de diverse tests. Dit kan de projectmanager zijn, vaak afkomstig uit of aangesteld namens de gebruikersorganisatie.

Opdrachtnemer

Degene die verantwoordelijk is voor het opstellen van het mastertestplan en/ of de uitvoering van de testopdracht. Deze persoon wordt in het algemeen aangeduid als de testmanager of overall testcoördinator.

Opdracht

De testmanager ondersteunt de opdrachtgever bij het formuleren van een heldere opdracht. Hierin dienen de doelstelling van het testtraject en een goede afbakening van de opdracht te worden aangegeven. Nadrukkelijk is de opdrachtformulering de verantwoordelijkheid van de opdrachtgever.

Voorbeeld 1

De opdracht aan het testproces omvat het testen van alle software (pakket en maatwerk) die het project XYZ gaat opleveren. De scope hiervan is in paragraaf x.x beschreven. De software moet zodanig worden getest dat er een solide personeelsadministratie met adequaat geconverteerde gegevens in productie genomen kan worden. Het testproces wordt ingedeeld volgens de in paragraaf y.y beschreven testsoorten en zodanig ingericht dat eventuele risico's vroegtijdig kunnen worden geïdentificeerd. Doel is om op basis van het inzicht in de kwaliteit van het op te leveren systeem een gefundeerd advies te geven ten aanzien van de in productie name.

Voorbeeld 2

Het mastertestplan is een aanvulling op het projectplan XYZ. Voor bijvoorbeeld projectbegrenzing, beschouwingsgebied, planning, enzovoort, wordt dan ook naar dit plan verwezen. Het project heeft als doel het compliant maken van werkplekondersteuning aan BDE. In simpele bewoordingen kan dit worden vertaald als:

- het geschikt maken van XYZ om onder Windows/XP te kunnen 'draaien';
- het regelen van de toegang tot werkplekondersteuning via BDE.

In dit mastertestplan wordt de relatie tussen de testsoorten systeemtest (ST), gebruikersacceptatietest (GAT) en de productieacceptatietest (PAT) beschreven. De testactiviteiten in de genoemde testsoorten moeten zo optimaal mogelijk op elkaar zijn afgestemd. Dit betekent dat onnodig, dubbel en/of niet testen van bepaalde objecten wordt voorkomen. De teststrategie is in overleg met projectleider XYZ en de lijnmanager Test&Support opgesteld. De relatie tussen de testsoorten is in dit mastertestplan op globaal niveau beschreven. Per testsoort werkt de daarvoor verantwoordelijke partij een afzonderlijk detailtestplan uit.

De globale testopdracht luidt:

- Rapporteer of, en zo ja welke, risico's <organisatie> loopt bij het in productie nemen van de XYZ release 2006 1.5 op basis van de uitgevoerde tests binnen de gekozen teststrategie.

Voorbeeld 3

Gebruik het project "Centralisatie SYS-VS" als pilot om de voorgestelde nieuwe testaanpak te toetsen voor de onderdelen zoals gedefinieerd in de scope. Blijf binnen de kaders van het PID (Project Initialisatie Document) tot de integratie en laat het overkoepelende project (SYS release 3.0) voor dit plan expliciet buiten beschouwing. Het project is planningstechnisch al bepaald; de aanpak moet daaraan ondergeschikt zijn. Onderdelen uit het plan zijn toe te passen als het een duidelijk toegevoegde waarde heeft op de directe, lopende activiteiten en overeenstemming daarover is bereikt, dit ter beoordeling aan de projectleider.

Voorbeeld 4

In de eerste fase van de Keteninformatisering worden verschillende kwaliteitszorgmaatregelen georganiseerd, waaronder toets- en testactiviteiten. Deze activiteiten moeten op tijd met inzicht in de gewenste kwaliteit door alle betrokken partijen zijn uitgevoerd.

Dit plan geeft de afspraken tussen de partners weer voor wat betreft de testen en toetsen die worden uitgevoerd en wie daarvoor verantwoordelijk zijn. Er wordt vastgesteld wat aan inzichten en rapporten geleverd wordt op basis waarvan besluitvorming kan plaatsvinden.

Naast de primaire opdracht bevat het mastertestplan soms ook een secundaire opdracht, bijvoorbeeld het verbeteren van de testprocessen van de betrokken testsoorten, zie ook bovenstaande voorbeeld 3. Aandachtspunt is dat de testmanager wel de eventueel benodigde extra tijd en middelen moet meenemen in de planning en begroting.

Beschouwinggebied

Hier dienen de grenzen gegeven te worden wat het beschouwinggebied van het testen is. Het gaat hierbij specifiek om de scope voor de uit te voeren testactiviteiten. Hierin moeten de volgende zaken worden meegenomen (indien van toepassing):

- syste(e)m(en);
- conversies;
- Administratieve Organisatie (AO-)procedures;
- interfaces met aangrenzende systemen (wordt de interface getest *tot* het andere systeem of *tot en met* of zelfs tot en met de hele keten?).

Daarnaast is van belang om aan te geven welke zaken buiten de scope van het testen vallen. Denk hierbij naast bovengenoemde zaken ook aan:

- systeemwijzigingen die niet in het project worden meegenomen (bijvoorbeeld hardware wijzigingen aan het mainframe platform);
- testactiviteiten die door andere partijen worden uitgevoerd;
- reorganisaties;
- mogelijk toekomstige projecten die op het huidige project van invloed zijn (met name als er over andere projecten nog geen duidelijkheid is).

Tips

- Voeg een plaatje toe dat het beschouwingsgebied visueel duidelijk maakt, door een lijn te trekken om de te beschouwen systemen, interfaces, AO, enzovoort.
- Overdrijf de lijst zaken die buiten de scope vallen niet. Het kan zeer verdedigend of vanzelfsprekend overkomen.

Randvoorwaarden

Onder randvoorwaarden worden voorwaarden beschreven die derden zoals bijvoorbeeld de opdrachtgever, het project of de gebruikers, aan het testproces opleggen en waarbinnen het testproces moet opereren.

Uitgangspunten

Dit zijn externe omstandigheden of gebeurtenissen die moeten gebeuren om het testproces succesvol te laten zijn, maar die buiten de controle van het testproces vallen. Anders gezegd: de eisen die het testproces stelt aan de anderen.

Producten

De opdrachtformulering, vastgelegd in het mastertestplan.

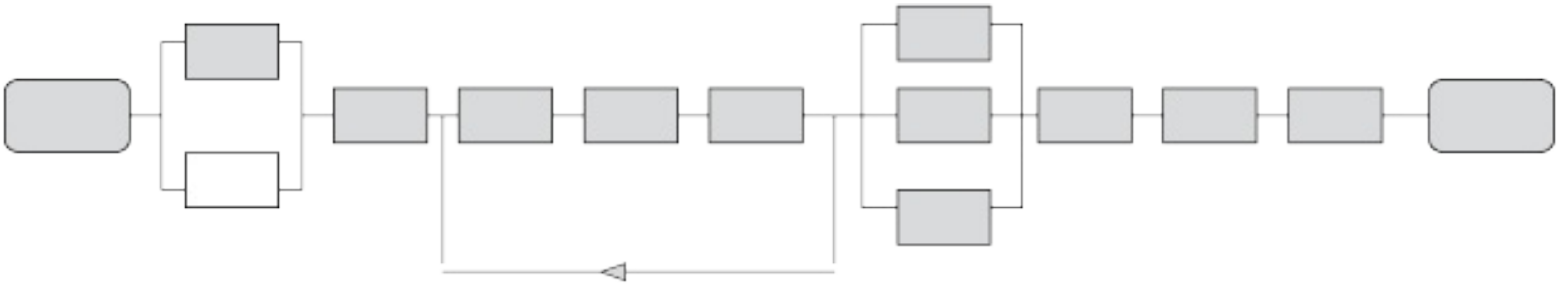
Technieken

Niet van toepassing.

Tools

Niet van toepassing.

5.2.2 Oriënteren opdracht



Doel

Het verkrijgen van inzicht in de (project)organisatie, de doelstelling en opzet van het systeemontwikkelproces, het te testen systeem of pakket en de eisen waaraan dit moet voldoen, zodat er beter richting gegeven kan worden aan de overige stappen van de planvorming.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen acceptanten met acceptatiecriteria en overige informatiegevers (zoals kwaliteitszorgmedewerker, materiedeskundigen, ontwerpers en systeembeheerders);
2. Het bestuderen van de beschikbare documentatie;
3. Het afnemen van interviews.

In de praktijk wordt deze activiteit parallel aan de opdrachtformulering uitgevoerd en ook enigszins onderschat. De onderschatting bestaat er met name uit dat de testmanager met te weinig acceptanten spreekt, terwijl het juist in het begin

essentieel is om de verwachtingen goed te peilen en als testmanager overal de “voelsprieten uit te steken”. Dit is nodig voor het goed kunnen uitvoeren van volgende activiteiten in de fase Planning én voor het in de toekomst goed kunnen beheersen van het totale testproces.

Ten opzichte van de gelijknamige activiteit in paragraaf 6.2.2 zijn de volgende aandachtspunten extra belangrijk:

- Het aantal acceptanten en betrokkenen voor het totale testproces is een stuk hoger dan voor een afzonderlijke testsoort én het moment ligt normaal gesproken vroeger in het traject, wanneer de verwachtingen het meest uiteenlopen. De testmanager moet een goed beeld zien te krijgen van deze verwachtingen en de politieke situatie. Waar liggen de gevoeligheden? Waar zit een spanningsveld? Draait het in dit project om tijd, geld of kwaliteit? Wat is het achterliggende belang voor de organisatie? Hoe zitten de betrokkenen bij het project erin? Wat zijn hun testdoelen?
- Het mastertestplan heeft relaties met diverse andere plannen in het ontwikkel- of onderhoudsproces, zoals al beschreven bij de context van het mastertestplan (paragraaf 3.2). De testmanager moet deze relaties inventariseren en er invulling aan geven vanuit het mastertestplan.
- Bij documentatie en interviews moet de testmanager extra aandacht geven aan het verzamelen van de testbasis, aangezien dit voor het mastertestplan niet als aparte activiteit wordt onderkend.

Vóórdat aan de volgende activiteit, Analyseren van de Productrisico’s, wordt begonnen, koppelt de testmanager de bevindingen van deze activiteit ter verificatie terug met de opdrachtgever.

Producten

Deze activiteit levert de onderstaande onderdelen van het mastertestplan. In het plan moet duidelijk aangegeven worden dat de genoemde zaken een eerste inventarisatie betreffen en dat in een later stadium, in de afzonderlijke testsoorten, dit verder uitgewerkt, geactualiseerd en gedetailleerd gaat worden:

- Acceptanten en acceptatiecriteria
  De voor het testen relevante acceptanten en hun acceptatiecriteria.
- (Verwijzing naar) Testbasis
  In het algemeen worden niet alle producteisen opgesomd in het (master)testplan. Er wordt volstaan met de verwijzing naar document(en) waarin de eisen zijn opgenomen. Het is daarbij belangrijk om ook een versienummer op te nemen, zodat steeds voor iedereen duidelijk is op welke versie(s) van document(en) de test is gebaseerd.
- Te hanteren normen en standaarden
  Hier worden de gebruikte standaarden genoemd. Wat betreft testen kan daarbij gedacht worden aan voorschriften van de lijnorganisatie Testen, aan TMap, aan het TPI-model [Koomen, 2006] of aan testhandboeken.
  Ontwikkelstandaarden, documentstandaarden of kwaliteitsnormen die gevolgd (moeten) worden, kunnen hier ook worden opgenomen.
- Gerelateerde documenten aan het mastertestplan
  Hier worden de documenten genoemd die een relatie hebben met dit mastertestplan. Denk hierbij aan een Projectplan of een Plan van Aanpak voor het project, specifieke project- of testplanningen, een specifieke of een generieke testmethode, een implementatieplan of andere documenten die van belang zijn.

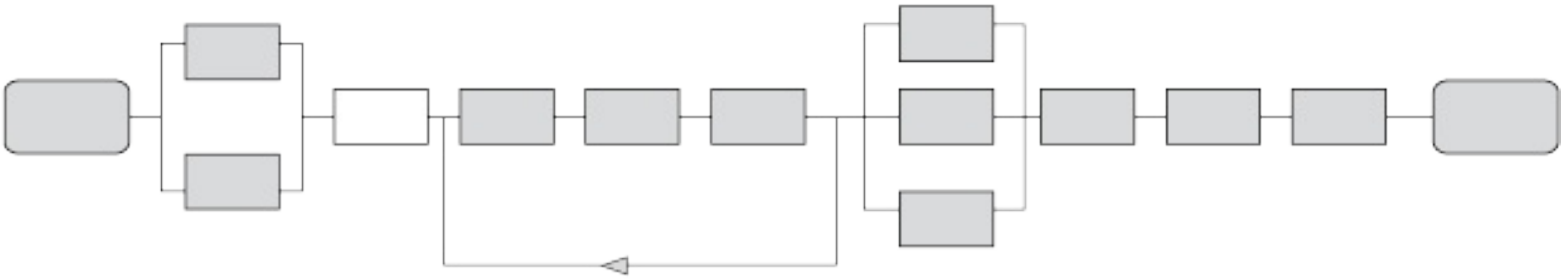
Technieken

Checklist “Oriënteren testopdracht” (www.tmap.net).

Tools

Niet van toepassing.

5.2.3 Analyseren Productrisico’s



Doel

De verschillende belanghebbenden en de testmanager komen tot een gezamenlijk beeld van wat de meer of minder risicovolle delen en kenmerken van het systeem zijn.

Werkwijze

Testen is een maatregel om inzicht te geven in de kwaliteit en daaraan gerelateerde productrisico's van een systeem of pakket bij in-productie-name voor een organisatie[1]. Aangezien er nooit sprake zal zijn van een onbeperkte hoeveelheid middelen en tijd is het belangrijk om zo vroeg mogelijk te bepalen welke systeemdelen en kenmerken extra of juist minder testinspanning vragen. Hiervoor moeten onderbouwd keuzes gemaakt worden. Hulpmiddel bij het bepalen van aandachtsgebieden voor de test is het uitvoeren van een productrisicoanalyse (PRA).

Het uitvoeren van een PRA valt uiteen in de volgende subactiviteiten:

1. Bepalen deelnemers
2. Bepalen van de PRA-aanpak
3. Voorbereiden sessie/interviews
4. Verzamelen en analyseren productrisico's
5. Volledigheidscontrole.

In hoofdstuk 9 "Productrisicoanalyse" worden deze stappen gedetailleerd toegelicht. Er moet rekening worden gehouden dat er voor een mastertestplan een grote variëteit aan deelnemers betrokken is, waardoor de risicoanalyse minder detail kan hebben. De deelnemers zijn vaak acceptanten maar ook overige informatiegevers, zoals genoemd in paragraaf 5.2.2. Zo zijn testdoelen vaak anders en is een indeling in deelobjecten verschillend voor technisch beheer en gebruikers en normaal gesproken afhankelijk van een kenmerk. Functionaliteit kent bijvoorbeeld een andere indeling van het systeem in deelobjecten dan performance of beveiliging. Voor kenmerken worden vaak kwaliteitsattributen gekozen, maar ook te testen aspecten kunnen als kenmerken worden gebruikt, zoals installeerbaarheid, multi-user, regressie, enzovoort. Aandachtspunt is dat de eventuele PRA-sessie mogelijk gecombineerd kan worden met (een deel van) de volgende activiteit, Bepalen teststrategie.

Producten

De risicotabellen met per kenmerk testdoelen en mogelijke deelobjecten met risico-indicaties, apart beheerd en optioneel vastgelegd in het mastertestplan.
Het PRA-totaaloverzicht met kenmerken/deelobjecten met risicoklasse, vastgelegd in het mastertestplan.

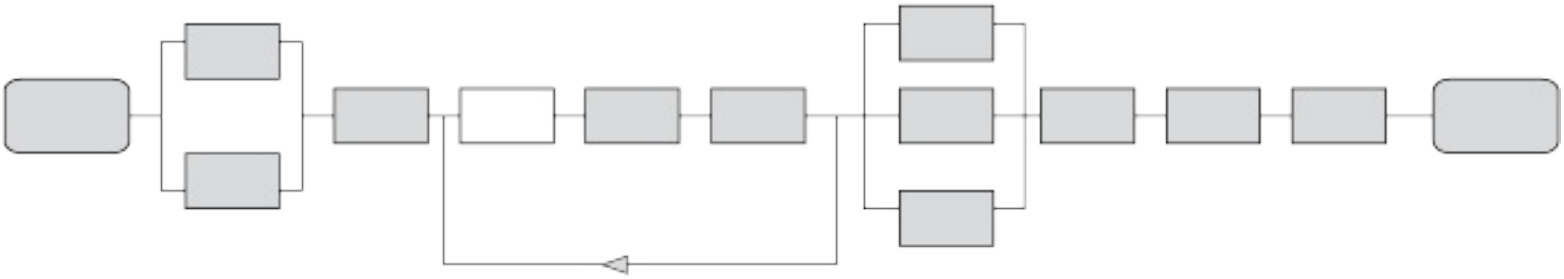
Technieken

Productrisicoanalyse (hoofdstuk 9).
Uitleg kwaliteitsattributen (paragraaf 10.2).

Tools

Niet van toepassing.

5.2.4 Bepalen teststrategie



Doel

Op basis van de productrisicoanalyse een keuze maken welk kenmerk/deelobject in welke testsoort hoe zwaar getest moet worden.

Werkwijze

Het opstellen van een teststrategie voor een mastertestplan bestaat uit de volgende activiteiten:

1. Bepalen testsoorten
2. Bepalen zwaarte van testen per kenmerk/deelobject per testsoort

Onderstaand worden deze activiteiten nader toegelicht.

De productrisicoanalyse en een eerste opzet van de teststrategie kunnen vaak gecombineerd worden in een enkele sessie. Lukt dit niet, dan maakt de testmanager na de productrisicoanalyse een voorstel voor de teststrategie en overlegt dit met opdrachtgever en een aantal andere belanghebbenden. Door gezamenlijk tot een teststrategie te komen, ontstaat een gedragen resultaat, waardoor in een later stadium beslissingen over de balans tussen testzwaarte, kosten en tijd beter en makkelijker gemaakt kunnen worden.

1. Bepalen testsoorten

Dit betreffen de testsoorten waarop het mastertestplan betrekking heeft. Er moeten goede afspraken gemaakt worden over welke testsoorten worden onderkend en welke partij verantwoordelijk is voor de uitvoering van welke testsoort. Veel organisaties maken onderscheid in een unittest (UT), unitintegratie test (UIT), systeemtest (ST), soms een functionele acceptatietest (FAT), een gebruikersacceptatietest (GAT) en een Productie-acceptatietest (PAT). De UT, UIT en ST vallen dan onder verantwoordelijkheid van de ontwikkelorganisatie. De FAT en GAT vallen onder verantwoordelijkheid van de toekomstige gebruikerorganisatie. De PAT is de verantwoordelijkheid van de toekomstige (technische) beheerorganisatie. Met grote voorkeur is toetsen (reviews) ook een onderdeel van het mastertestplan, zoals veel organisaties al doen. Dit is met name het geval in organisaties waarin geen apart kwaliteits- of toetsplan wordt opgesteld.

Veel organisaties hebben een standaard indeling van testsoorten. De testmanager moet onderzoeken in hoeverre hiervan afgeweken kan of moet worden.

Als de testsoorten nog niet vastgesteld zijn of als er afwijkingen mogelijk zijn, onderzoekt de testmanager in deze stap wat de gewenste indeling in testsoorten is, rekening houdend met de resultaten van de productrisicoanalyse.

Bij het bepalen van de testsoorten worden afspraken gemaakt over het doel en de inhoud van elke testsoort.

Uitgediept

Overwegingen voor indeling testsoorten

- Gekozen/gehanteerde ontwikkelmethode
  Diverse ontwikkelmethoden vullen het testen op geheel eigen wijze in. Dit kan een afweging zijn om de testsoorten uit zo'n methode simpelweg te adopteren.
- Organisatiestructuur van de afdelingen
  Als een organisatie een functionele afdelingenstructuur kent (productieafdeling, gebruikersafdeling, beheerafdeling, enz.) kan het een overweging zijn om elke betrokken afdeling onafhankelijk van de andere afdelingen zijn tests te laten plannen enz. Dit kan zich uiten in bijvoorbeeld de testsoorten productieacceptatietest,

gebruikersacceptatietest en/of functionele acceptatietest.

- Gescheiden verantwoordelijkheden
  Een leverende partij heeft andere (test)verantwoordelijkheden dan de accepterende partij. Binnen deze twee partijen zijn verdere opdelingen vaak voor de hand liggend. Als bijvoorbeeld een productieafdeling verantwoordelijk wordt gehouden voor een stabiele exploitatie van een systeem is het te overwegen om de testsoort productieaccepatietest te benoemen. Het kan nog specifieker als bijvoorbeeld deze afdeling ook verantwoordelijk wordt gehouden voor een bepaalde performance van het systeem. In dat geval is het te overwegen om dan de testvorm performancetest als testsoort te organiseren. Ook wanneer componenten of services van derden worden afgenomen, bijvoorbeeld bij service georiënteerde architecturen, is aan te bevelen om de acceptatie van deze componenten of services als aparte testsoort te organiseren.
- Diverse belanghebbenden
  In de situatie dat een belanghebbende zeker wil weten dat er aan zijn eisen/wensen is voldaan kan daarvoor een aparte testsoort worden benoemd. Bijvoorbeeld bij een pakketimplementatie, waarbij de gebruiker niet overtuigd is dat het pakket alle gewenste functionaliteiten bezit. Dan is het te overwegen om de testsoort functionele (acceptatie)test te benoemen.
- Mate van risico
  Bij hoge risico's is het benoemen van een aparte testsoort te overwegen om het testen van het risico voldoende focus te geven. Bijvoorbeeld in het geval van een systeem met vele (risicovolle) interfaces kan een testsoort systeemintegratietest een goede aanvulling zijn. Of als bijvoorbeeld beveiliging of performance grote risico's vormen, kan overwogen worden om die testvormen als een aparte testsoort te organiseren (securitytest, performancetest).
- Test-/ontwikkelvolwassenheid
  Als deze volwassenheden laag zijn is het vaak verstandig om testsoorten niet te combineren of te integreren. In zo'n situatie zie je vaak de testsoorten systeemtest (uitgevoerd door leverende partij) en functionele test (uitgevoerd door accepterende partij) apart benoemd worden. Is de volwassenheid daarentegen hoog, dan zou je deze tests juist kunnen combineren tot een zogenaamde gecombineerde test (GT), waarbij de doelen van beide tests gehandhaafd blijven, maar de organisatie en uitvoering gecombineerd worden.
- Contractuele afspraken
  In een demand-supply situatie moet een leverende partij aan contractuele afspra ken voldoen. Dit kan in de vorm zijn dat de partij in een systeemtest moet aantonen dat aan bepaalde requirements is voldaan, of door het uitvoeren van een securitytest dat een internetapplicatie niet "te kraken" is, of dat tijdens een performancetest wordt aangetoond dat het pakket ook met productiedata een acceptabele responsetijd heeft. Zeker wanneer ook sprake is van certificering (bij smart cards, bij security voor webapplicaties, bij compliancyregelingen als SOX) is aan te bevelen deze als aparte testsoort in te richten.
- Beschikbaarheid infrastructuur
  Soms wordt men gedwongen een bepaalde indeling in testsoorten te herzien, om dat simpelweg bepaalde testinfrastructuren beperkt benaderbaar zijn. Zo kan er bijvoorbeeld een aparte systeemintegratietest worden benoemd, omdat een externe partij hun testomgeving maar beperkt gedurende een bepaalde periode wil openstellen voor het uitvoeren van tests ("testwindow").
- Mate van integratie
  In algemene zin zou men kunnen zeggen dat als de complexiteit van (de omgeving van) een systeem toeneemt het te overwegen is om de tests op te delen in meerdere testsoorten om het geheel beheersbaar te maken en te houden. Verder is het mogelijk dat sommige kwaliteitsattributen pas getest kunnen worden als het systeem een zekere mate van integratie heeft bereikt. Een voorbeeld hierbij is performance. Dit zou je het liefst zo vroeg mogelijk willen meten, maar vaak kan dat pas realistisch worden gemeten als het systeem zo goed als af is. Dit zou een overweging kunnen zijn om hiervoor een aparte testsoort te benoemen.
- Afname of acceptatie
  Om een onderscheid te maken tussen afname en acceptatie zou je kunnen zeggen dat het er bij de afname vooral om gaat of er aan de afgesproken eisen is voldaan en aan de plicht om te betalen (zie ook de overweging contractuele afspraken). Bij de acceptatie gaat het juist vooral om de vraag: "Kan de organisatie hiermee werken?".
  In dat geval spelen bijvoorbeeld ook de niet beschreven eisen een rol.

De volgende afwijkingen komen in de praktijk regelmatig voor:

- Ketentest
  Een ketentest vraagt afstemming tussen diverse systemen, partijen, infrastructuren, enzovoort. Dit is bij veel organisaties dusdanig complex dat de test als aparte testsoort (systeemintegratietest), wordt ingericht, met eigen testmanager en testers.

- Gecombineerde test
  De systeemtest en (functionele) acceptatietest hebben inhoudelijk meestal hetzelfde doel, namelijk testen of het systeem functioneel correct werkt. Het verschil is de verantwoordelijke partij: leverancier of afnemer. Uit efficiëntieoogpunt kunnen deze testsoorten gecombineerd worden tot een enkele gecombineerde test, mits goede afspraken zijn gemaakt over taken en verantwoordelijkheden, beheer en controle op correcte uitvoering.
- Securitytest, performancetest, usabilitytest
  Als een testvorm een zeer specifieke testomgeving of expertise behoeft, kan het zinvol zijn deze als aparte testsoort te organiseren, met een eigen plan en organisatie. Dergelijke tests worden daarom ook vaak uitbesteed.
- Proof-of-concept test
  Bij veel onzekerheid, dus wanneer het erg moeilijk is de risico's goed in te schatten, wordt soms een zogenaamde proof-of-concept test ingericht. Bij grote migraties zien we deze testsoort regelmatig. In de proof-of-concept test wordt beproefd of de voorgestelde project- en/of ontwikkelaanpak gaat werken. Dit gebeurt door een klein (zo'n 10%) maar representatief deel van het systeem in een vroeg stadium te bouwen en te testen. De resultaten en opgedane ervaringen geven meer zekerheid over het totale proces en helpen het geheel te optimaliseren.

Tips

- Inflexibele standaard indeling testsoorten
  Zoals al genoemd hanteren veel organisaties een standaard indeling van de testsoorten. Aanpassen hiervan kost in sommige gevallen buitensporig veel energie of is zelfs een onhaalbare kaart. In die gevallen doet de testmanager er beter aan de uit te voeren tests te plooien onder de bestaande testsoorten.
- Uitvoeringsvolgorde testsoorten
  Een indeling in testsoorten is niet per definitie de volgorde van uitvoeren van tests.
  Een geplande performancetest in een productieacceptatietest (PAT) kan soms al heel vroeg worden uitgevoerd, wellicht al voor de gebruikersacceptatietest of net na of zelfs parallel aan de systeemtest. M.a.w., al wordt de PAT als laatste in het rijtje testsoorten genoemd, dan betekent dat nog niet dat deze als laatste moet worden uitgevoerd. Dit wordt bepaald bij het opstellen van de planning (paragraaf 5.2.6) en is ook afhankelijk van de af te dekken risico's.

2. Bepalen zwaarte van testen per kenmerk/deelobject per testsoort

Na het bepalen van de testsoorten die in de scope van het mastertestplan meegenomen worden, worden de onderkende kenmerken/deelobjecten uit de productrisicoanalyse toegewezen aan deze testsoorten. Op deze wijze wordt een afstemming bereikt tussen de diverse testsoorten die binnen het project worden uitgevoerd. Uiteraard dienen hierbij de diverse verantwoordelijkheden en bevoegdheden niet uit het oog te worden verloren. Er kan meestal al een soort standaardverdeling gehanteerd worden, gebaseerd op het doel van de testsoorten (ST voornamelijk functionaliteit, GAT inpasbaarheid en gebruikersvriendelijkheid, enzovoort).

Uitgediept

Zoals eerder gesteld kan ook het toetsen onder het beschouwingsgebied van het mastertestplan vallen en dus ook onder de strategiebepaling. Toetsen is alleen mogelijk op de documentatie die (formeel) opgeleverd wordt. Documentatie kan omvatten: ontwerpen op allerlei niveaus (requirements, use cases, functioneel ontwerp, procesontwerp, technisch ontwerp, AO-procedures), gespreksverslagen, resultaten van brainstormsessies, enzovoort.

Als eerste worden de kenmerken en deelobjecten uit de PRA, met risicoklasse (hoog, gemiddeld, laag), uitgezet tegen de testsoorten. Vervolgens wordt voor elke kenmerk/deelobject-combinatie bepaald hoe zwaar en in welke testsoort deze getest moet worden. Door middel van een ● teken wordt in een matrix aangegeven of een bepaald kenmerk/deelobject onderdeel uitmaakt van de teststrategie van een bepaalde testsoort. Met ● ● of zelfs ● ● ● wordt aangegeven dat er relatief veel aandacht dient te worden gegeven aan het kenmerk/deelobject tijdens de betreffende testsoort. Uiteraard kan een kenmerk/deelobject tijdens meerdere testsoorten worden meegenomen, maar de diepgang zal veelal verschillen. Indien gebruik wordt gemaakt van testontwerptechnieken, kan bijvoorbeeld de acceptatietest de resultaten van voorafgaande testsoorten inzien op basis waarvan eventueel met een mindere diepgang kan worden getest.

Kenmerk/deelobject	PRA-RK	Toetsen	Ontwikkeltest	ST	GAT	PAT

Functionaliteit						
- deelsys 1	A	●	● ●	● ●	● ●	
- deelsys 2	B	●	●	● ●		
- totaal	C			●	●	
Gebr.vriendelijkheid	B	●		I	● ●	
Performance						
- online	B			●		●
- batch	C					●
Beveiliging	A	●		S	●	● ● ●
Inpasbaarheid	B	●			● ● ●	

Toelichting bij de bovenstaande tabel:

PRA-RK	Risicoklasse (uit PRA met A= hoog risico, B= gemiddeld risico, C= laag risico)
Toetsen	Toetsing/review van de verschillende tussenproducten, zoals requirements, functioneel ontwerp, technisch ontwerp
Ontwikkeltest	Unittest + Unitintegratietest
ST	Systeemtest
GAT	Gebruikersacceptatietest
PAT	Productieacceptatietest
●	beperkte dynamische test
● ●	gemiddelde dynamische test
● ● ●	zware dynamische test
S	Statisch testen Controleren en onderzoeken van producten zonder dat de software wordt uitgevoerd.
I	Impliciet testen Meetesten bij een andere testvorm zonder het maken van specifiek hiervoor ontworpen testgevallen.
<leeg>	Als in een cel niets staat, betekent dit dat de betreffende toets- of testsoort geen aandacht hoeft te besteden aan het kenmerk.

Tip

De strategiebepaling is meer dan het mechanisch toekennen van testzwaarte-bolletjes op basis van de risicoklassen. Andere aspecten die erbij horen zijn:

- het goed indelen van de te hanteren testsoorten;
- het betrekken van toetsen/reviews, zodat al in een vroeg stadium bevindingen gedaan worden die dan nog relatief goedkoop te herstellen zijn;
- verfijnde afbakening van het totale testen en van de individuele testsoorten, zodat duidelijk is wat tot het beschouwingsgebied van elke testsoort hoort;
- niet-testers hebben moeite met de interpretatie van een dergelijke tabel. Schrijf daarom bij de tabel een voor iedereen begrijpelijke toelichting (wat wordt getest, met wat voor diepgang of dekkingsgraad, hoe wordt overlap voorkomen, enzovoort).

Uitgediept

Overlap in tests

Een belangrijk doel van het toewijzen van kenmerken/deelobjecten aan testsoorten is ervoor te zorgen dat er niet onbedoeld tests dubbel of niet worden uitgevoerd. Vanzelfsprekend kan er in bepaalde situaties bewust voor gekozen worden om vergelijkbare tests door meerdere partijen te laten uitvoeren. Dit is bijvoorbeeld zinvol als een bepaald testaspect vanuit verschillende kanten belicht kan worden. Zo kan bijvoorbeeld het kenmerk beveiliging zowel in de GAT als in de PAT worden meegenomen. In de GAT worden dan de autorisaties getest, terwijl in de PAT gekeken wordt naar de technische werking van de firewalls.

Incrementen

Bij iteratieve of agile systeemontwikkeling werken de ontwikkelaars met incrementen, een soort tussenreleases. Elk increment bevat meer functionaliteit. De testmanager kan hier bij de strategiebepaling rekening mee houden door één globale mastertestplanstrategie op te stellen met de algemene aanpak voor testen, waarna de afzonderlijke testsoorten steeds de strategie bepalen per increment. Alternatief is dat de testmanager voor elk increment een kleine (mastertestplan)strategiebepaling doet in plaats van één grote.

Tips

- Met name voor functionaliteit: een zwaarte van ● ● ● voor een bepaalde test suggereert dat in de testsoort het gehele (deel)systeem met de zwaarst mogelijke diepgang getest wordt. Dat is niet het geval! Het aantal bolletjes geeft aan dat er zwaarder, gelijk of juist lichter getest moet worden dan gemiddeld. Het is daarmee een opdracht aan de testmanager van de betreffende testsoort om dit verder uit te werken in de detailteststrategie.
- De testmanager moet bij het opstellen van de strategie al zoveel mogelijk rekening houden met de afweging van kosten, tijd en benodigde vaardigheden. Als hij weet dat er maar zeer beperkt budget is of dat de beschikbare mensen geen ervaring hebben met testen, moet het voorstellen van “onmogelijke” strategieën zoals heel dure tests of zeer grondige testontwerptechnieken voorkomen worden (om teveel terugkoppeling-slagen te vermijden).
- Uitgangspunt is dat een test zo vroeg mogelijk in het systeemontwikkeltraject moet worden uitgevoerd. Dit verlaagt de herstelkosten. Afhankelijk van de gewenste afdekking van een risico kan men bijvoorbeeld denken aan een vroege usabilitytest in de ontwikkelomgeving. Niet ieder kenmerk kan echter in een vroegtijdig stadium getest worden. Bijvoorbeeld het testen van inpasbaarheid vereist de aanwezigheid van een (bijna) compleet systeem.

Producten

De teststrategie, inclusief toelichting, vastgelegd in het mastertestplan.
Korte beschrijving per testsoort, met minimaal het doel van de test en de verantwoordelijke persoon of afdeling.

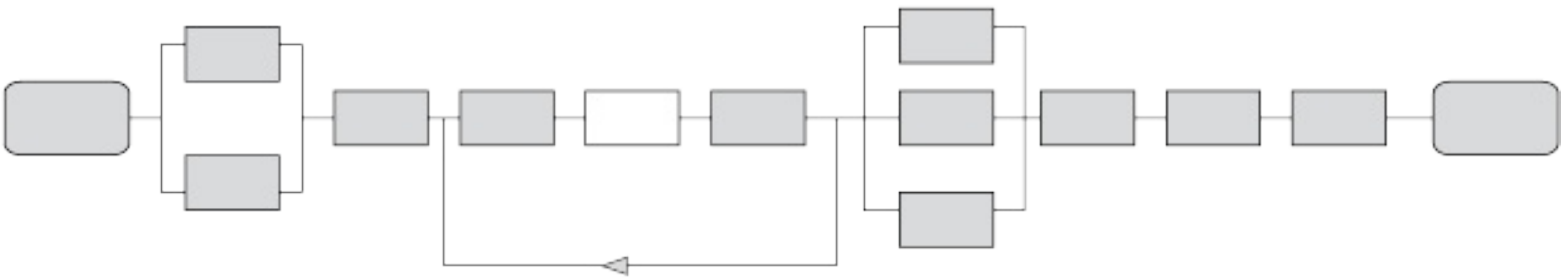
Technieken

Strategiebepaling (zoals beschreven in deze paragraaf).

Tools

Niet van toepassing.

5.2.5 Bepalen begroting



Doel

Gebaseerd op de teststrategie een globale begroting opstellen voor het totale testproces zodat de opdrachtgever deze kan goedkeuren of bijstelling kan vragen.

Werkwijze

Het opstellen van een begroting voor een mastertestplan gebeurt vroeg in het project en is gebaseerd op de opgestelde teststrategie. Veelal is dan nog niet alle kennis over het testobject aanwezig. Dit heeft als consequentie dat de nauwkeurigheid van de begroting beperkt is. De omvang en/of complexiteit van het testobject kan gedurende het project nog veranderen. Ook vormen de testomgevingen en eventuele testtools een (forse) kostenpost. Het is belangrijk dat de testmanager aan de belanghebbenden duidelijk maakt dat de begroting gebaseerd is op een aantal aannames en daarom later verder gedetailleerd en mogelijk herzien moet worden. Een mogelijke oplossing hiervoor is om marges te gebruiken om de initiële begroting voor een mastertestplan weer te geven.

De begroting in het mastertestplan vormt het kader voor de begrotingen per testsoort (bijvoorbeeld systeemtest, gebruikersacceptatietest en productieacceptatietest).

Uitgediept

Begrotingstechnieken

Met name het kiezen van de begrotingstechnieken is een stap waarvoor ervaring vereist is. Er bestaan voor het opstellen van een begroting verschillende begrotingstechnieken. Deze begrotingstechnieken en de stappen om tot een begroting te komen staan beschreven in hoofdstuk 11 “Begrotingstechnieken”. Voor een mastertestplan kan worden begroot op basis van:

- verhoudingsgetallen
- testobjectomvang
- work breakdown structure (WBS)
- proportioneel begroten
- testpuntanalyse (TPA).

Om de betrouwbaarheid van de begroting te vergroten is het gebruik van zowel eigen ervaringscijfers als meerdere begrotingsmanieren en -technieken sterk aan te bevelen.

Producten

De globale begroting voor het totale testproces, inclusief aannames, vastgelegd in het mastertestplan.

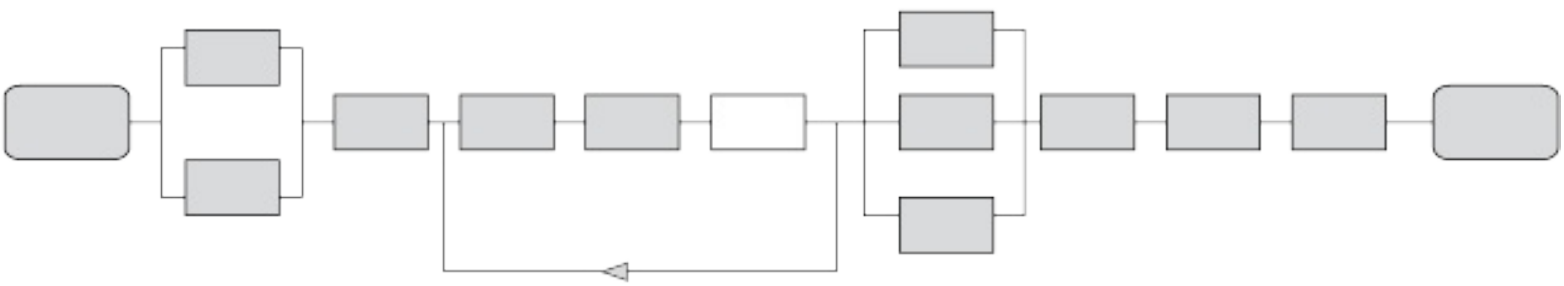
Technieken

Diverse begrotingstechnieken inclusief TestPuntAnalyse (hoofdstuk 11).
Stappenplan voor opstellen begroting (paragraaf 11.1).

Tools

Plannings- en voortgangsbewakingstool (spreadsheets voor het begroten van de testinspanning en voor TPA zijn verkrijgbaar op www.tmap.net).

5.2.6 Bepalen planning



Doel

Het opstellen van een globale, zo betrouwbaar mogelijke planning voor alle testsoorten die tot de scope behoren, zodat de opdrachtgever hier rekening mee kan houden of kan bijsturen. Uitgangspunt van de planning is om de belangrijkste fouten het eerst te vinden, binnen het kader van de strategie en begroting.

Werkwijze

Aan de hand van de planning van het systeemontwikkelproces, de teststrategie en begroting wordt een globale planning voor het totale testproces opgesteld. Ook factoren die bij het oriënteren zijn gebleken, zoals of het een tijd, geld of kwaliteit gestuurd project is, beïnvloeden de planning. Per testsoort worden de start- en einddatum en de op te leveren producten aangegeven. In de fase Planning van de diverse testsoorten wordt de detailplanning uitgewerkt.

De globale planning dient minimaal te omvatten:

- uit te voeren activiteiten (op faseniveau per testsoort);
- relaties met en afhankelijkheden van andere activiteiten (binnen of buiten het testproces en tussen de diverse testsoorten);
- te besteden tijd per testsoort;
- benodigde en beschikbare resources (mensen en infrastructuur);
- benodigde en beschikbare doorlooptijd;
- op te leveren producten.

Als de opdrachtgever dit wenst, moeten de financiële gevolgen van de gemaakte keuzen zichtbaar worden gemaakt in een financiële planning. Denk hierbij aan kosten voor (intern en extern) personeel, training, werkplekken, testomgevingen en -tools.

Het streven moet zijn om de testuitvoeringsactiviteiten van de verschillende testsoorten op elkaar aan te laten sluiten of gecontroleerd te laten overlappen (dakpansgewijs testen). Wanneer omwille van de mijlpalen bepaalde planningskeuzes gemaakt worden die risico's met zich meebrengen, moet de testmanager dit melden en toelichten.

Een aan kwaliteit gerelateerd aspect van planning is wanneer een testsoort klaar is. De testmanager heeft daarom een belangrijke rol bij het afstemmen van de entry- en exitcriteria van de opvolgende testsoorten. Voor uitleg over de relatie tussen acceptatiecriteria en exitcriteria, zie paragraaf 6.2.7 "Bepalen planning".

Voorbeeld

Exit-/acceptatiecriteria

In dit praktijkvoorbeeld hebben exit- en acceptatiecriteria grote overlap.

De testaanpak en de acceptatiecriteria zijn afgestemd met de betrokken stakeholders (zie paragraaf x.y). Er zijn twee niveaus van acceptatie te onderscheiden:

1. Acceptatie van systeem XYZ door alle betrokken stakeholders. Dit betreft het vrijgeven van de producten voor productie.
2. Individuele acceptatie van elk uitgevoerde testsoort door de opdrachtgever(s) van deze testsoort.

Voor **niveau 1** acceptatie geldt dat de tests conform de gezamenlijk afgesproken teststrategie zijn uitgevoerd en dat voor eventueel aanwezige bevindingen aan onderstaande richtlijnen is voldaan: De producten van XYZ kunnen in productie (zijn geaccepteerd) als:

- Er geen bevindingen van *categorie A* aanwezig zijn.
- Er voor bevindingen van *categorie B* een "patch" of "workaround" beschikbaar is.
  Hierbij moet de ontwikkelaar tevens een planning hebben afgegeven, waarin wordt vermeld hoe en wanneer het probleem structureel wordt opgelost.
- Er voor bevindingen van *categorie C* een document beschikbaar is, waarin wordt vermeld hoe met deze non-critical bevindingen wordt omgegaan.

Voor **niveau 2** acceptatie geldt dat aan het testteam (verantwoordelijk voor de desbetreffende testsoort) decharge wordt verleend als de doelen, zoals globaal verwoord in paragraaf y.z en nader zijn gespecificeerd in eventuele detailtestplannen, zijn gehaald. Dit is tevens een Go/NoGo beslissing voor het starten van de testuitvoering van de

volgende testsoort.

Terugkoppeling

De testmanager koppelt de combinatie van gekozen teststrategie, begroting en planning terug met de opdrachtgever voor goedkeuring of bijstelling, waarbij de opdrachtgever een keuze kan maken voor betere risicoafdekking tegen meer testtijd en geld of juist andersom. De voorgaande stappen voor Strategie, Begroting en Planning herhalen zich zo nodig. Om de communicatie te vergemakkelijken grijpt de testmanager hierbij weer terug naar de oorspronkelijke testdoelen.

Daarnaast bespreekt de testmanager met opdrachtgever het hanteren van toleranties bij het uitvoeren van het testproces. Dit zijn grenzen waarbinnen de testmanager de opdrachtgever geen toestemming hoeft te vragen. Zo wordt vaak een tolerantie van 5% voor de begroting aangehouden.

Producten

De globale planning voor het totale testproces, vastgelegd in het mastertestplan.
Entry- en exitcriteria per testsoort.
Met de opdrachtgever teruggekoppelde strategie, begroting en planning.
Optioneel: toleranties voor strategie, begroting en planning.

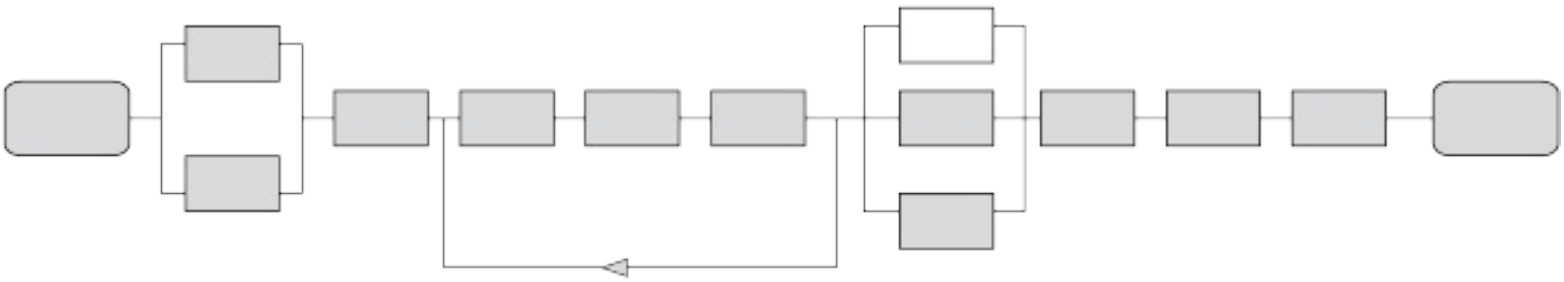
Technieken

Niet van toepassing.

Tools

Plannings- en voortgangsbewakingstool.
Workflowtool.

5.2.7 Definiëren testproducten



Doel

Het definiëren van de op te leveren testproducten op overkoepelend niveau en door de verschillende testsoorten.

Werkwijze

De activiteiten die uitgevoerd worden voor het plannen en beheersen van het totale testproces leveren bepaalde producten op zoals het mastertestplan en rapportages, maar ook procedures, voorschriften en projectdocumentatie als overlegnotulen. In overleg met opdrachtgever en andere betrokkenen wordt bepaald wat de op te leveren producten zijn. Vervolgens kan in het mastertestplan een keuze gemaakt worden welke testsoorten welke testproducten moeten opleveren. Dit kan gaan om testware zoals testplannen of testscripts of (geautomatiseerde) regressietests, dus producten die voor hergebruik in aanmerking komen, maar ook om testdocumentatie als voortgangsrapportages. Naast het benoemen van op te leveren producten kunnen hier ook normen en standaarden gegeven worden, waar de producten aan moeten voldoen. Tenslotte helpt het gemeenschappelijk gebruik van tools voor testwarebeheer- of planning en

voortgangsbewaking om een uniforme werkwijze af te dwingen en het latere beheer te vergemakkelijken.

Uitgediept

Waarom op het niveau van mastertestplan?

Er zijn redenen om het afstemmen van testproducten in het mastertestplan op te nemen in plaats van dit aan elke testsoort over te laten:

- het dwingt een uniforme werkwijze af bij testsoorten waardoor een betere beheersing van het totale testproces mogelijk is. Met name wanneer externe partijen bepaalde tests uitvoeren, is dit een belangrijk argument;
- het vergemakkelijkt de communicatie tussen testsoorten en testmanagement;
- de herbruikbaarheid van de producten wordt vergroot, niet alleen binnen een testsoort maar ook over de testsoorten heen;
- de uniforme werkwijze maakt het makkelijker mensen tussen testsoorten te laten rouleren.

Producten

Een beschrijving van de (per testsoort en op overkoepelend niveau) op te leveren testproducten inclusief normen en standaarden, vastgelegd in het mastertestplan.
Sjablonen voor de diverse documenten.

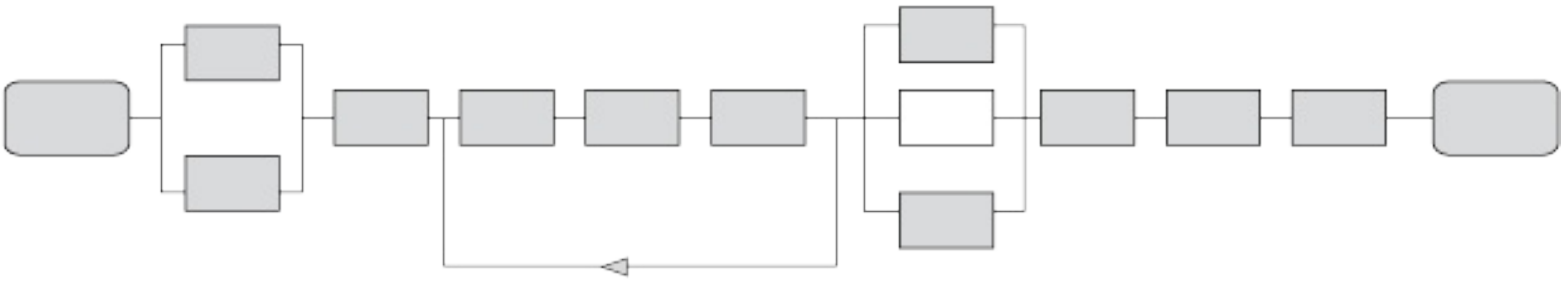
Technieken

Niet van toepassing.

Tools

Testwarebeheertool.

5.2.8 Definiëren organisatie



Doel

Het definiëren van de rollen, taken, bevoegdheden en verantwoordelijkheden die van toepassing zijn voor het totale testproces, over de testsoorten heen.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen benodigde rollen;
2. Toewijzen taken, bevoegdheden en verantwoordelijkheden;
3. Beschrijven organisatie;
4. Toewijzen personeel;
5. Vaststellen opleidingen en coachingsbehoefte;

6. Vaststellen overleg- en rapportagestructuren.

1. Bepalen benodigde rollen

Om een goede afstemming tussen de diverse testsoorten te laten plaatsvinden, wordt globaal bepaald welke testrollen (zie hoofdstuk 16 “Testrollen”) hiervoor moeten worden onderkend en ingevuld. Denk hierbij met name aan:

- Testmanagement of testcoördinatie;
- Beheer (op mastertestplan niveau, denk hierbij bijvoorbeeld aan beheer van de testinfrastructuur of de bevindingenadministratie voor alle testsoorten).
- Ondersteuning (op mastertestplan niveau, denk hierbij bijvoorbeeld aan testtoolspecialisten, methodische testspecialist voor alle testsoorten, materiedeskundigen of ontwikkelaars)

Ten behoeve van de resourceclaim vindt tevens een globale bepaling plaats van de benodigde testrollen per testsoort. Dit wordt later verder ingevuld tijdens de fase Planning van de betreffende testsoorten.

2. Toewijzen taken, bevoegdheden en verantwoordelijkheden

Hier worden per benodigde rol de taken en verantwoordelijkheden weergegeven. Dit is nog niet op testsoortniveau (dit wordt beschreven in de detailtestplannen), maar op overkoepelend niveau.

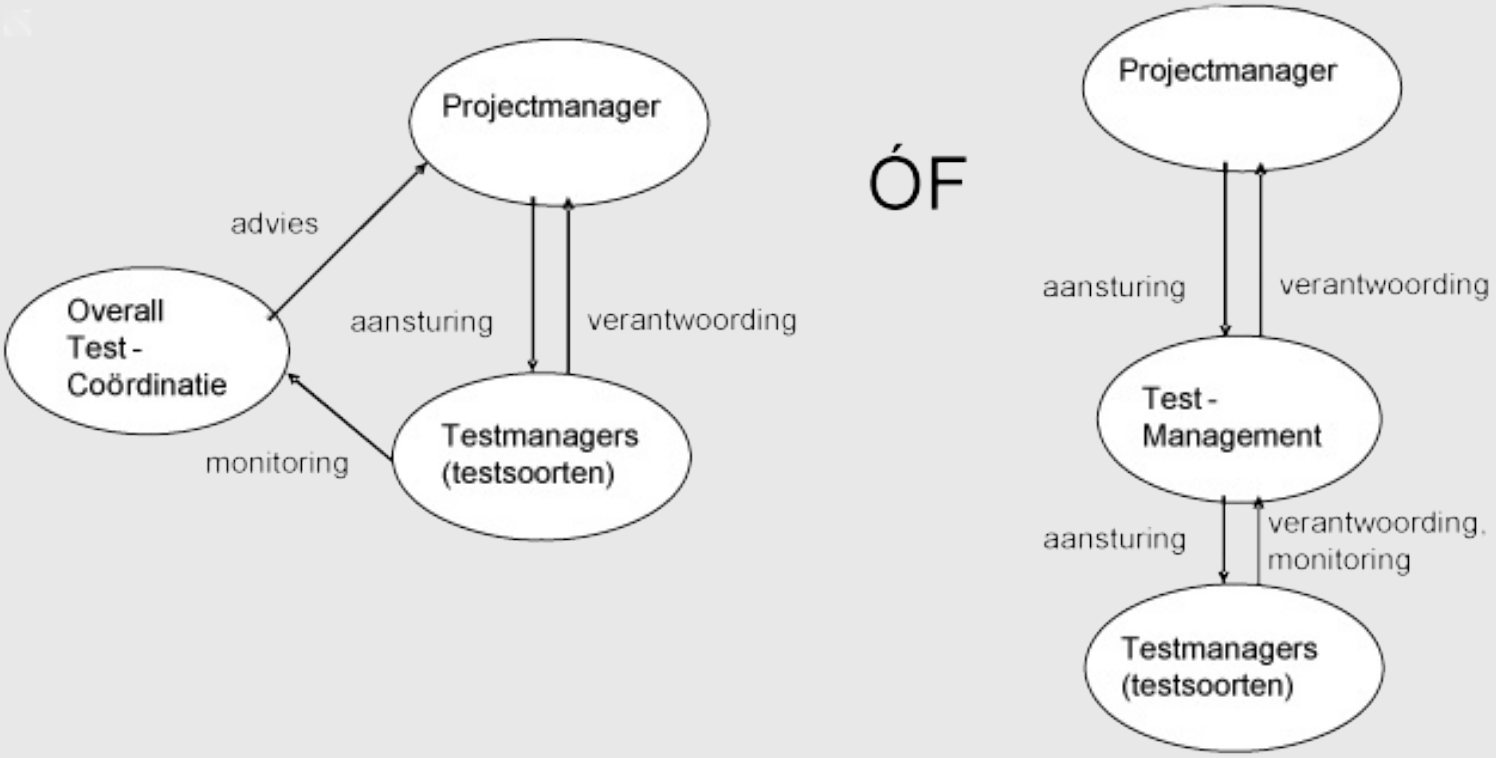
Uitgediept

Voorbeelden van taken

- het opstellen van de voorschriften voor de producten die door de diverse testsoorten worden opgeleverd;
- de controle op naleving van de voorschriftgeving (interne reviews);
- het coördineren van de verschillende testactiviteiten die voor de diverse testsoorten vergelijkbaar zijn, zoals het inrichten van de technische infrastructuur en het beheer ervan;
- het opstellen van richtlijnen voor de communicatie en rapportage tussen de testsoorten onderling en het testproces en de diverse leveranciers;
- het inrichten van overkoepelende testmethodische, technische en functionele ondersteuning;
- het consistent houden van de diverse testplannen;
- het rapporteren over de totale testvoortgang en kwaliteit van het testobject;
- de inzet/inhuur van (extra) testpersoneel.

Uitgediept

Manager of coördinator?



Figuur 5.4: Mogelijke verantwoordelijkheden testcoördinatie of -management.

In de praktijk is het met name belangrijk om de verantwoordelijkheid van testmanagement en -coördinatie scherp te krijgen. Anders gezegd: wanneer een bepaalde testsoort onbeheersbaar is geworden en de verantwoordelijke testmanager niet in staat is om dit recht te trekken, is de (overkoepelende) testmanager hiervoor dan (eind)verantwoordelijk of is zijn/haar rol meer coördinerend en adviserend? In het eerste geval is testmanager de goede term, in het tweede geval is het meer een overall testcoördinator. In geval van het eerste betekent dit wel dat de testmanager ook de bijbehorende bevoegdheden moet hebben. Denk hierbij aan het hebben van eigen budget en het kunnen nemen van maatregelen, zoals het inzetten van extra mensen of zelfs het vervangen ervan. In het tweede geval omvat de taak het coördineren van de verschillende tests, het voorschrijven en toezicht houden op de manier van testen en het informeren en adviseren van het projectmanagement over de testvoortgang en de kwaliteit van het geteste systeem. In de praktijk zien we dat er meestal sprake is van overall testcoördinatie omdat testen vaak bij meerdere partijen belegd is. Testmanagement zien we met name wanneer diverse testsoorten bij één partij liggen.

Overigens moet opgemerkt worden dat in de praktijk vaak mengvormen tussen management en overall coördinatie voorkomen, bijvoorbeeld dat de testmanager rechtstreeks één of meer testsoorten aanstuurt (bijvoorbeeld de ST en de GAT), en de coördinatie voert over andere testsoorten (bijvoorbeeld de ontwikkeltests en de PAT).

Het tekenen van het organisatieplaatje kan helpen de situatie helder te maken.

3. Beschrijven organisatie

De samenhang tussen de genoemde rollen, de afzonderlijke testsoorten en de relaties met de andere betrokken partijen binnen het systeem ontwikkelproces moet bepaald en vastgelegd worden. De organisatie van het testen maakt vanzelfsprekend onderdeel uit van het grotere (project)geheel. In het geval van een project moet niet vergeten worden ook de relatie naar een eventuele test- of kwaliteitsafdeling te leggen.

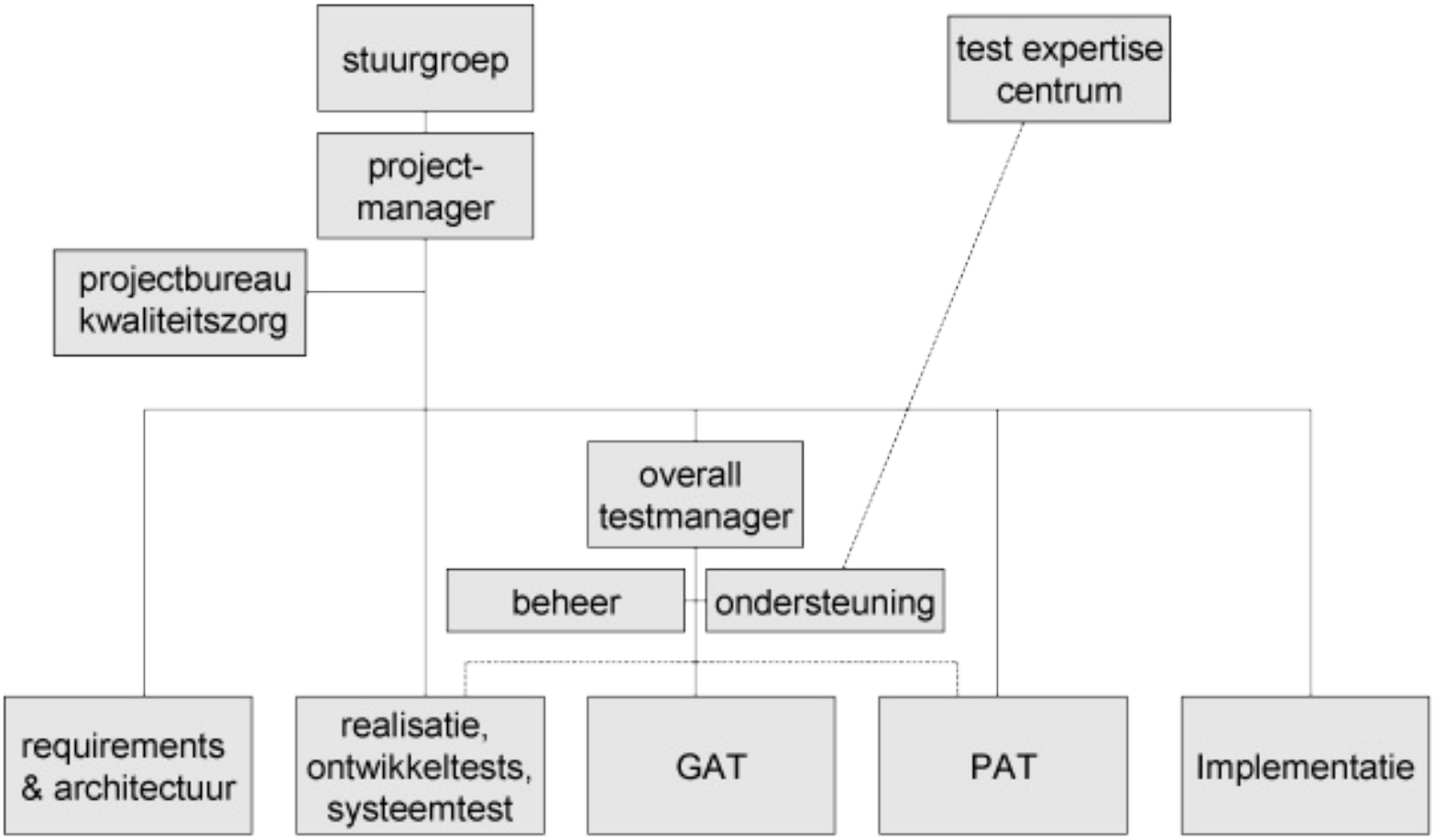
In het kader van een mastertestplan moet het testen georganiseerd worden, zowel op overkoepelend niveau als op testsoort-niveau. Voor de organisatie van een testsoort zijn in grote lijnen de volgende mogelijkheden te onderkennen:

1. testen als **zelfstandige activiteit** of **geïntegreerd met andere activiteiten**;
2. testen in een **project-** of in een **lijnorganisatie** belegd;

Omdat deze keuzes afhankelijk zijn van de testsoort, project en organisatie en er op mastertestplan-niveau vaak maar weinig invloed op uitgeoefend kan worden, wordt voor verdere toelichting verwezen naar paragraaf 6.2 “Fase Planning” en naar paragraaf 8.3 “Permanente testorganisatie”.

Daarnaast is de keuze om het overkoepelende management, beheer en ondersteuning onder te brengen in een project- of lijnorganisatie.

Figuur 5.5 toont een voorbeeld van een vrij traditionele organisatiestructuur.



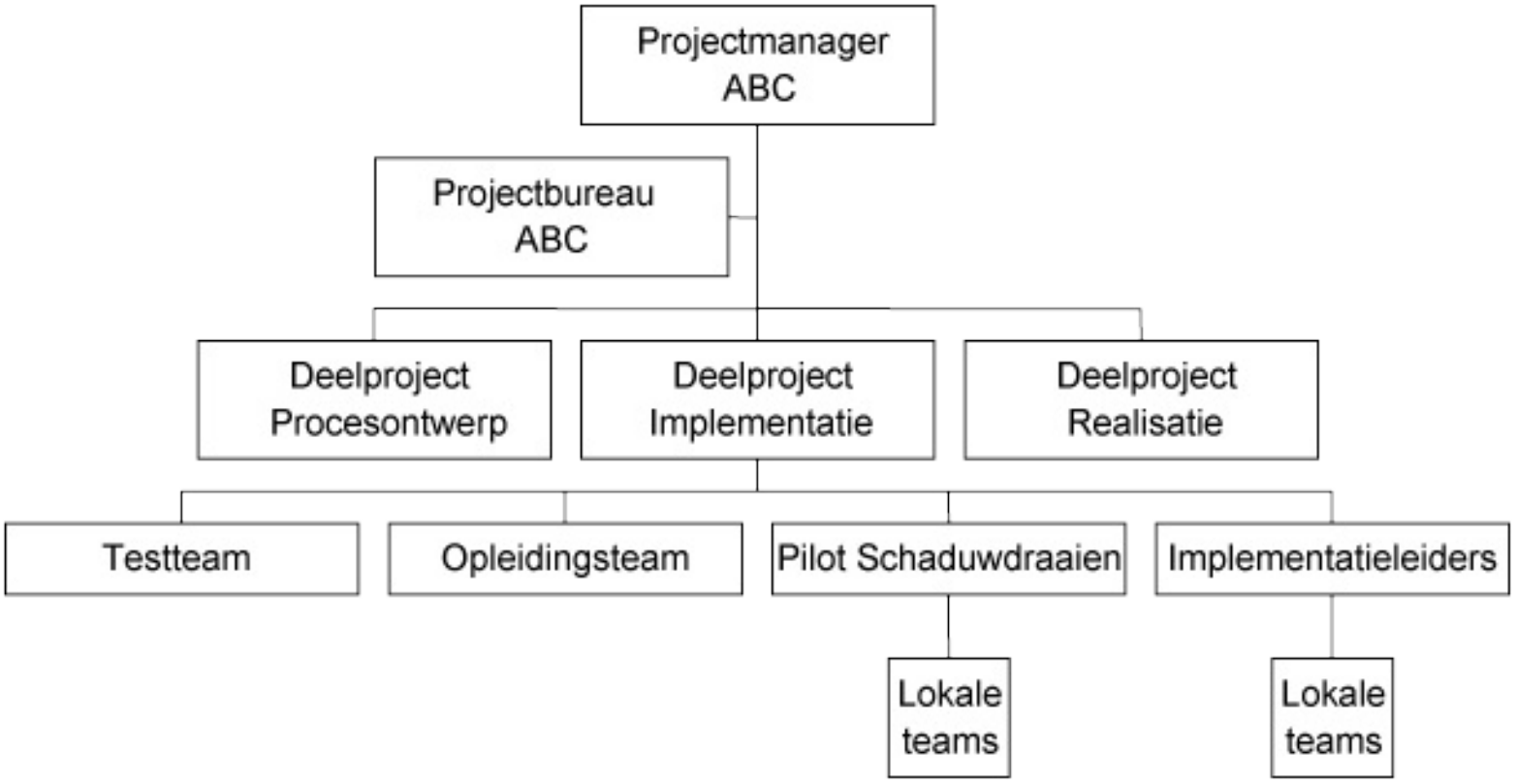
Figuur 5.5: Voorbeeld organisatie van testen.

Dit voorbeeld heeft een mengvorm van testmanagement en testcoördinatie.

De testmanager stuurt rechtstreeks de GAT aan, maar heeft enkel een coordinerende taak naar de andere testsoorten zoals de ontwikkeltests, de systeemtest en de PAT. Overkoepelende ondersteuning en beheer voor testen vinden plaats onder de vlag van testmanagement, waarbij er een relatie is met de lijnorganisatie Testen, het test expertise centrum, voor gebruik van generieke aanpak, normen, standaarden, sjablonen en tools. De testmanager valt onder de projectmanager. Deze is verantwoordelijk voor het invullen van de uitgangspunten voor het testproces, zoals het tijdig beschikbaar komen van personeel, middelen, planningen en testbasis. Natuurlijk besteedt de projectmanager veel van dit regelwerk uit aan de testmanager. De relaties met de opdrachtgever van het project en de stuurgroep worden onderhouden door het projectmanagement.

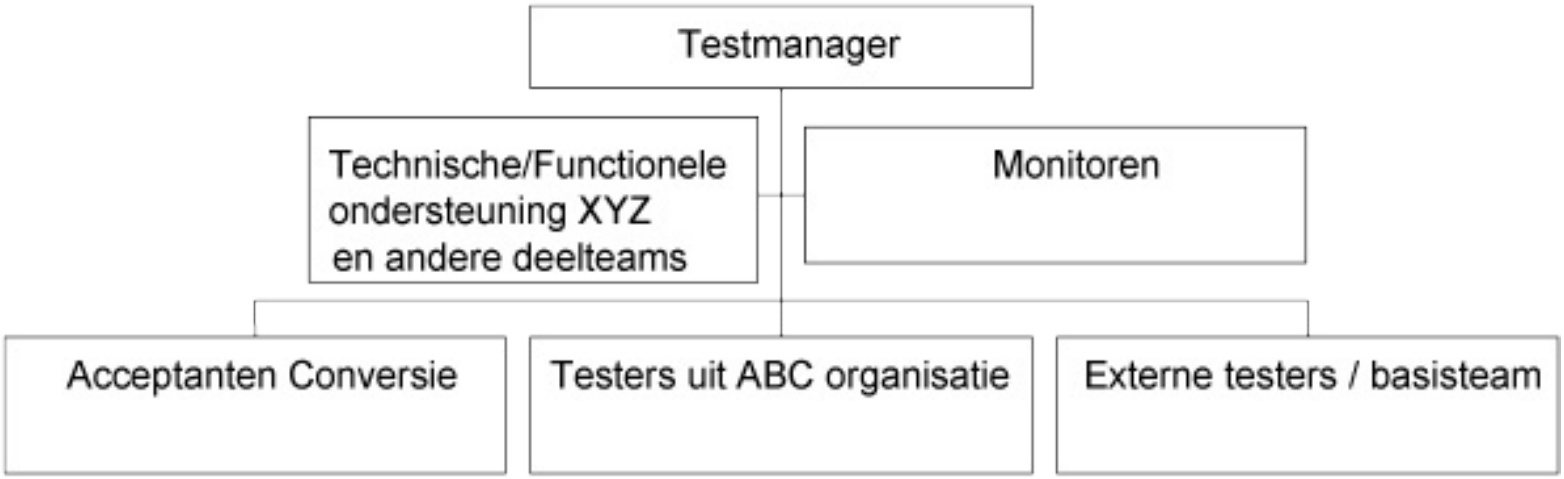
Op verzoek wordt soms periodiek of bij afronding van de tests rechtstreeks aan de stuurgroep en/of aan de opdrachtgever advies uitgebracht over de kwaliteit.

Een ander voorbeeld van een organisatie betreft een pakketimplementatie:



Figuur 5.6: Voorbeeld organisatie van testen pakketimplementatie.

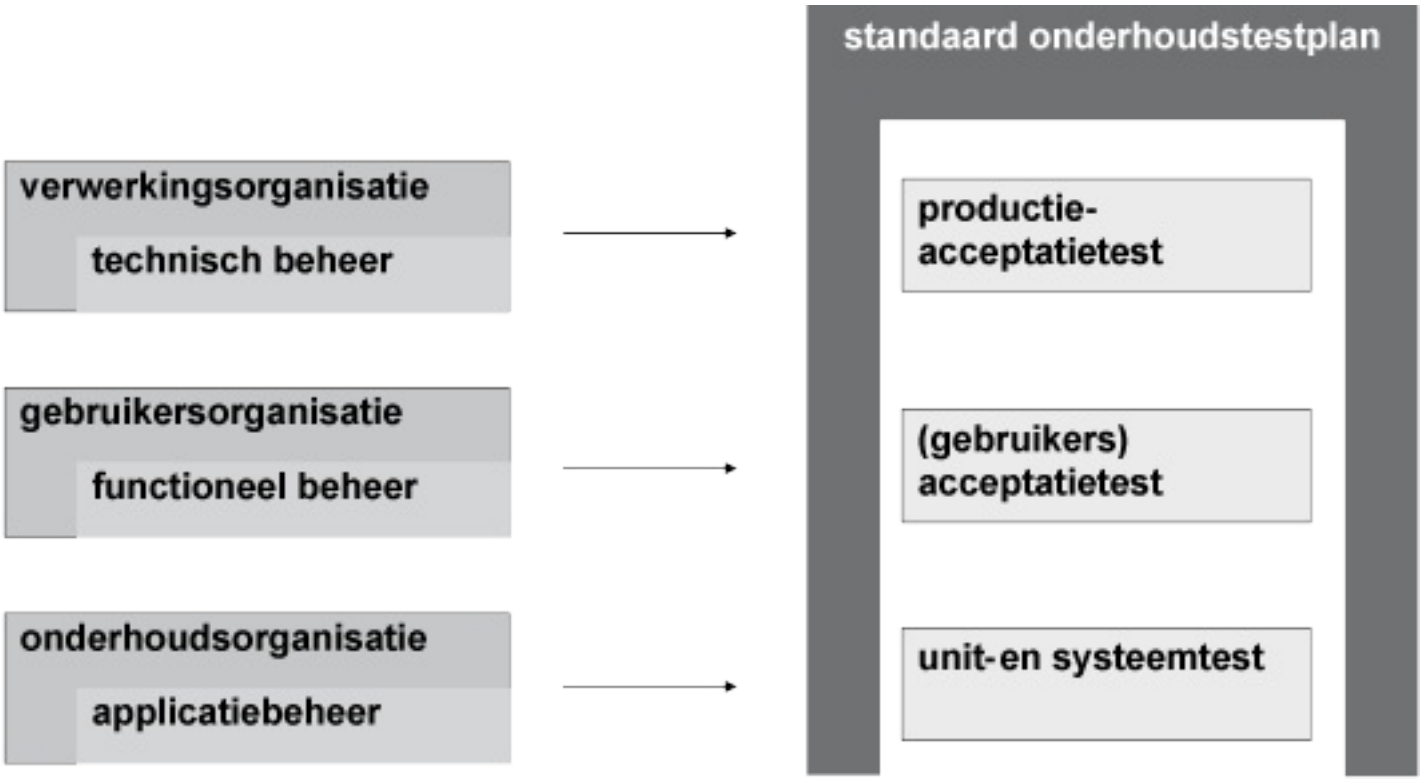
Bovenstaand testteam is als volgt samengesteld:



Figuur 5.7: Voorbeeld testteam pakketimplementatie.

Voor het testen is een testteam samengesteld, onder leiding van een testmanager. De testmanager is verantwoordelijk voor het volledige testtraject. Het team wordt flexibel samengesteld voor de diverse activiteiten, maar heeft een vaste kernbezetting. Het testproces wordt gedurende het gehele testtraject gemonitord door representanten van de accepterende partijen. Deze representanten nemen deel aan het tweewekelijkse monitorenoverleg. De monitoren hebben te allen tijden toegang tot alle documentatie uit het testtraject. Door het continu monitoren kan het testproces, indien noodzakelijk geacht door de monitoren, in een vroeg stadium worden aangepast aan de eisen van de monitorende partij. Te denken valt hier bijvoorbeeld aan eisen van de toekomstige beheerorganisatie voor het testen van autorisaties en integratietests.

Bij onderhoud kunnen de testsoorten worden benoemd en wie, of welke afdelingen, voor de invulling en uitvoering daarvan verantwoordelijk zijn. In algemene zin kan men zeggen dat de *gebruikersorganisatie* verantwoordelijk is voor de (gebruikers)acceptatietest, de *verwerkingsorganisatie* voor de productieacceptatietest en de *onderhoudsorganisatie* voor de unit-/systeemtest (zie onderstaande figuur).



Figuur 5.8: Relatie beheervormen met testsoorten.

Uitgediept

Iteratieve en agile systeemontwikkeling

Bij deze vormen van systeemontwikkeling is een belangrijke vraag wie verantwoordelijk is voor een testsoort. Bij de unit- en unitintegratietests is dit snel duidelijk, maar bij de systeem- en acceptatietests is er een aantal mogelijkheden. In onderstaande tabel staan de meest voorkomende testsoorten uitgezet tegen de verantwoordelijke partijen.

Testsoort	Welke partij?
Unittest	Ontwikkeling (binnen team)
Unitintegratietest	Ontwikkeling (binnen team)
Systeemtest	Ontwikkeling (binnen team)
(Functionele / gebruikers-) acceptatietest	Gebruikersorganisatie (binnen of buiten het team)
Productieacceptatietest	Systeembeheerorganisatie (meestal buiten het team)

Met “binnen of buiten het team” wordt bedoeld dat een test kan worden uitgevoerd binnen het projectteam, maar dat het voor sommige testsoorten ook mogelijk is dat een afzonderlijk testteam buiten het projectteam of zelfs buiten de organisatie wordt ingericht om deze test uit te voeren. Zo wordt er meestal voor gekozen om de productieacceptatietest buiten het projectteam uit te voeren. Deze test vereist specifieke (systeembeheer) expertise van de testers plus een testomgeving die zo productie getrouw mogelijk is. Beide zijn vaak maar beperkt in het project aanwezig.

De sterke betrokkenheid van beslissingsbevoegde gebruikers en de samenwerking binnen het team maken dat de traditionele muur tussen ontwikkelaars en gebruikers/ acceptanten een stuk lager is. Gebruikers geven gedurende het traject continu feedback op prototypes (door middel van toetsen en testen) waardoor de acceptatietest een hele ‘dunne’ test kan zijn. Deze zal dan vooral bestaan uit tests die niet eerder konden worden uitgevoerd, zoals bijvoorbeeld een ketentest. Indien de acceptatietest wordt uitgevoerd binnen het project, is dan ook te overwegen deze te integreren met de systeemtest. Het testteam bestaat dan uit een mix van gebruikers, ontwikkelaars en professionele testers.

Minder voor de hand liggend is om de acceptatietest buiten het project uit te voeren. Feitelijk is dit niet conform de principes van iteratief ontwikkelen. Welke vorm gekozen moet worden, is afhankelijk van de risico’s. Voorziet de organisatie grote risico’s, dan is een acceptatietest buiten het team meer voor de hand liggend, ook al doet dit afbreuk aan de iteratieve principes. Voorbeelden van dergelijke risico’s zijn dat de gebruikers onvoldoende beslissingsbevoegd zijn, dat de projectmanager of ontwikkelaars de gebruikers lijken te “overheersen” of dat het

systeem dermate kritisch is dat het testen plaats dient te vinden in een voldoende beheerste, stabiele en als-warehet-productieomgeving.

4. Toewijzen personeel

Nadat is vastgesteld welke testrollen er binnen het testproces vervuld moeten worden, worden mensen toegekend aan elke rol. Hierbij houdt men uiteraard rekening met hun beschikbaarheid en vaardigheden in relatie tot de voor de betreffende testrollen vereiste kennis en vaardigheden (zie paragraaf 8.6 “Testprofessionals” en hoofdstuk 16 “Testrollen”). Overigens kan één persoon meerdere rollen vervullen, dit komt met name veel voor in iteratieve ontwikkelomgevingen. Bij meerdere rollen voor dezelfde persoon moet opgelet worden dat dit niet tot conflicterende verantwoordelijkheden leidt!

5. Vaststellen opleidingen en coachingsbehoefte

De mensen die betrokken worden bij de testsoorten moeten verschillende soorten kennis hebben, namelijk op het gebied van testen, de materie en het systeem.

Op overkoepelend niveau moet men met name denken aan het organiseren van opleidingen voor betrokkenen uit meerdere testsoorten. Ook is het aan te bevelen om de diverse indirect betrokkenen bij het testproces, zoals project- of lijnmanagement, via een presentatie voor te lichten over het belang van (gestructureerd) testen.

6. Vaststellen overleg- en rapportagestructuren

Vanuit het totale testproces moet met verschillende partijen gecommuniceerd worden. Met elke partij moet afgesproken worden of overleg en/of rapportage plaatsvindt, en wat doel en frequentie ervan zijn.

Overlegvormen

Voor overlegvormen wordt afgesproken wie aanwezig zijn en wat eventueel de standaard agenda is.

Voorbeelden van overlegvormen voor de testmanager zijn:

- wekelijks overleg met alle testmanagers en/of testcoördinatoren;
- wekelijks projectoverleg;
- periodiek stuurgroepoverleg.

Rapportages

Rapportage kan op diverse manieren plaatsvinden, naar verschillende doelgroepen en op verschillende momenten. Afhankelijk van het soort rapportage is deze gebaseerd op één of meer van de BDTM-aspecten Resultaat, Risico’s, Tijd en Kosten. De belangrijkste vormen van rapportage zijn:

- voortgangsrapportage
- risicorapportage
- vrijgaveadvies
- eindrapportage

Van elk van deze rapportagevormen bepaalt de testmanager wie welke rapportage naar wie verstuurt, met welke inhoud en mate van detail, en met welke frequentie. In de activiteit “Oriënteren opdracht” heeft de testmanager al gekeken welke partijen rapportages moeten/willen ontvangen. In overleg met de opdrachtgever wordt dat nu nader ingevuld. De rapportagevormen en -inhoud worden gedetailleerd besproken in paragraaf 6.3.3 “Rapporteren”.

Uitgediept

Voorbeelden zijn:

- wekelijkse rapportage per testsoort over voortgang van testen en kwaliteit en risico’s van het testobject, geleverd

door de testmanager van een testsoort aan de testmanager en de opdrachtgever en andere betrokkenen (zoals het projectoverleg);
- wekelijkse samenvattende/aanvullende rapportage over voortgang van het totale testproces en kwaliteit en risico's van het testobject, op basis van bovenstaande rapportages, van de testmanager aan de opdrachtgever en andere betrokkenen (zoals het projectoverleg);
- periodieke rapportage over voortgang van het totale testproces en kwaliteit en risico's van het testobject, van de testmanager aan het stuurgroepoverleg.

Inhoudelijk is het voortgangs- en kwaliteitsrapport het meest van belang, omdat op basis van deze informatie en adviezen nog tijdig (bij)gestuurd kan worden. De gegevens hiervoor worden aangeleverd door (de rapportages van) de testsoorten en door beheer in te richten op overkoepelend niveau, zie ook paragraaf 5.2.10 "Inrichten beheer".

Producten

Een beschrijving van de testorganisatie, vastgelegd in het mastertestplan.

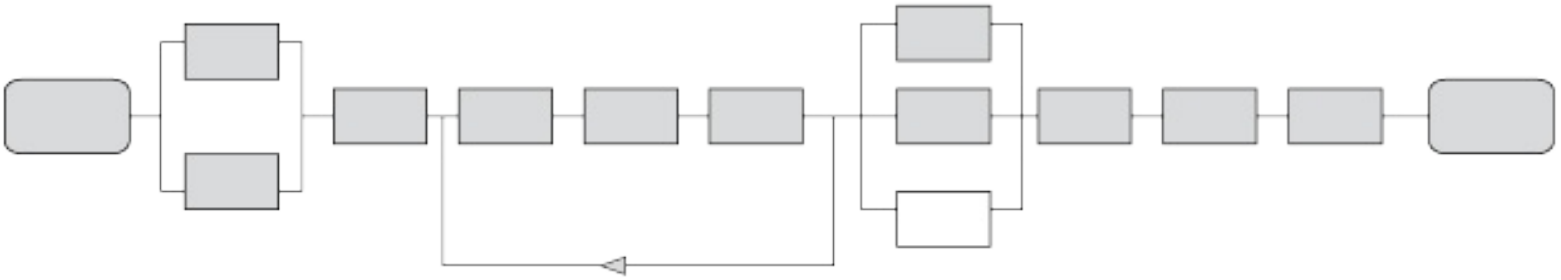
Technieken

Niet van toepassing.

Tools

Niet van toepassing.

5.2.9 Definiëren infrastructuur



Doel

Het in een vroegtijdig stadium vaststellen van de voor het testproces benodigde infrastructuur, met name de gedeelten die voor meerdere testsoorten moeten worden ingericht of een relatief lange besteltijd hebben.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Definiëren testomgeving;
2. Definiëren testtools;
3. Definiëren kantoorinrichting;
4. Vaststellen infrastructuurplanning.

Een uitgebreide beschrijving van testomgevingen en -tools is te vinden in respectievelijk paragraaf 8.4 "Testomgevingen" en 8.5 "Testtools".

1. Definiëren testomgeving

Elke testsoort heeft een testomgeving nodig om de tests uit te voeren. Deze omgeving is op hoofdlijnen samengesteld uit

de volgende componenten:

- hardware
- software
- communicatiemiddelen
- faciliteiten voor opbouw en gebruik van bestanden
- procedures.

De omgeving moet dusdanig worden samengesteld en ingericht dat aan de hand van de testresultaten optimaal kan worden vastgesteld in hoeverre het testobject aan de gestelde eisen voldoet. De omgeving heeft grote invloed op de kwaliteit, doorlooptijd en kosten van het testproces.

Om de omgeving goed te kunnen beheersen, is deze dan ook vaak apart van de ontwikkel- of productieomgeving. Hierbij stelt elke testsoort andere eisen aan de testomgeving. De vuistregel hierbij is dat de vroege tests, zoals de ontwikkeltests, meer behoefte hebben aan een flexibele, makkelijk aan te passen omgeving. Latere tests als de acceptatietest hebben juist meer behoefte aan stabiliteit en representativiteit. Zo worden de ontwikkeltests meestal in de Ontwikkelomgeving uitgevoerd, de systeemtest in een eigen Testomgeving, de acceptatietests in een Acceptatieomgeving, vlak voor de eigenlijke Productieomgeving. Dit wordt ook wel het OTAP-model genoemd (zie paragraaf 8.4.5 “OTAP-model”).

Op www.tmap.net is een checklist te vinden die behulpzaam kan zijn bij het definiëren van de testomgeving.

Wanneer de testomgevingen al bestaan, bijvoorbeeld in een onderhoudstraject, kan volstaan worden met hiernaar te verwijzen en de eventuele te maken aanpassingen te benoemen.

Uitgediept

Hoewel elke testsoort de detailinvulling van de voor haar benodigde omgeving kan regelen, zijn er toch redenen om er al op het niveau van het mastertestplan mee te beginnen:

- Het regelen, voorbereiden en inrichten van een testomgeving kost vaak veel tijd, het kan daarom het beste al zo vroeg mogelijk in gang gezet worden.
- Een omgeving is (erg) duur, er is binnen een testsoort niet altijd budget voor een eigen omgeving. Beperken van het aantal testomgevingen door de omgeving te delen met een andere testsoort kan dan een optie zijn. Dit vraagt goede afstemming en planning.
- De kosten van de testomgevingen moeten meegenomen worden in de totale begroting van het mastertestplan.
- Een andere optie die soms gekozen wordt, is om bijvoorbeeld de acceptatieomgeving na afloop van de test tot productieomgeving te maken. Nadeel hiervan is dat wanneer het systeem eenmaal in productie is, er geen omgeving meer is voor acceptatietesten van wijzigingen of nieuwe releases. Ook hier kan vanuit het mastertestplan afstemming plaatsvinden.
- Vaak krijgt een testsoort standaard één testomgeving. Hierbij wordt geen rekening gehouden dat binnen een testsoort vaak allerlei kwaliteitsattributen getest worden, waarvan het testen heel andere eisen aan de omgeving stelt. Zo is voor een performancetest een als-ware-het-productie omgeving nodig, maar een test van gebruikersvriendelijkheid kan misschien beter in een prototype-achtige omgeving plaatsvinden of in een usabilitylab met videocamera’s en doorzichtige spiegelwanden. Dit vraagt vroegtijdig regelen van de testomgevingen.
- Ook wordt de uitvoering van testsoorten als sequentieel gezien, “na de systeemtest van versie 0.5.2 kan deze versie naar de acceptatieomgeving worden overgezet”. Deze sequentiële uitvoering houdt onvoldoende rekening met de behoefte om tests zo vroeg mogelijk uit te voeren. Zo kan een test op gebruikersvriendelijkheid van de schermen vanuit de gebruikersacceptatietest al plaatsvinden wanneer de systeemtest hier nog geen functionele test op heeft uitgevoerd. Dit kan dan ofwel in de ontwikkelomgeving, ofwel in de acceptatieomgeving, mits deze zo vroeg al beschikbaar is. De afweging tussen het verkrijgen van eerdere testresultaten en een mindere representativiteit wordt hierbij bewust gemaakt. Andersom kunnen de ontwikkelaars performance als hoog risico inschatten en een (vroege) performancetest willen uitvoeren. Hiervoor is de als-ware-het-productie acceptatieomgeving dan het meest geschikt. Dit vraagt vooraf en vroeg afstemming.
- Met alle testsoorten en hun omgevingen kan er al gauw een flink aantal testomgevingen ontstaan. De vraag is wie deze omgevingen beheert. Dit kan een afdeling systeembeheer of infrastructuur zijn, of een lijnorganisatie Testen. Voor goede communicatie moet deze beherende partij een (vast) aanspreekpunt hebben binnen het testproces. Dit kan de testmanager zijn, maar het is beter om een (test)infrastructuurspecialist op overkoepelend niveau in het testproces te hebben.

2. Definiëren testtools

De benodigde testtools worden in grote lijnen gedefinieerd. Testtools kunnen ondersteuning bieden bij de meeste testactiviteiten.

Voor het mastertestplan ligt de focus op de tools die voor meerdere testsoorten gebruikt kunnen worden. Voorbeelden hiervan zijn tools voor testwarebeheer, geautomatiseerde testuitvoering en bevindingenbeheer en simulatoren.

3. Definiëren kantoorinrichting

De voor testen benodigde kantoorinfrastructuur (werkkamers, vergaderkamers, telefoons, PC's, netwerkaansluitingen, kantoorsoftware, printers, enzovoort) wordt in grote lijnen gedefinieerd. Het gaat hierbij om kantoorinrichting in de breedste zin, want ook de testers moeten hun werk onder goede omstandigheden kunnen uitvoeren. Een checklist voor de kantoorinrichting is te vinden op www.tmap.net.

4. Vaststellen infrastructuurplanning

Voor alle benodigde onderdelen van de infrastructuur wordt vastgesteld wie verantwoordelijk is voor de verdere uitwerking, selectie en verwerving. De gemaakte afspraken worden vastgelegd. Tevens wordt een globale planning opgesteld met daarin de tijdstippen van het ter beschikking stellen van de diverse faciliteiten.

Producten

De beschrijving van de benodigde infrastructuur op overkoepelend niveau, inclusief planning, vastgelegd in het mastertestplan.

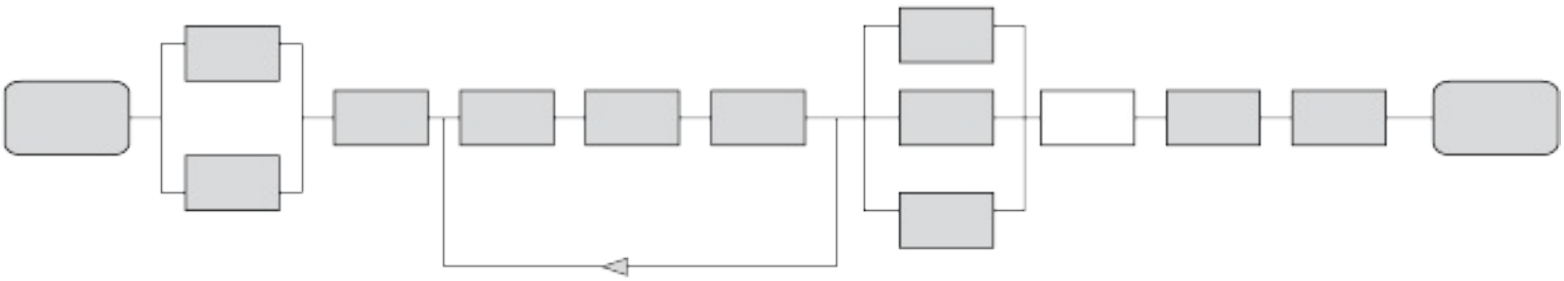
Technieken

Checklist "Testomgevingen" (www.tmap.net).
Checklist "Kantoorinrichting" (www.tmap.net).

Tools

Niet van toepassing.

5.2.10 Inrichten beheer



Doel

Het vastleggen van de wijze waarop het beheer van het testproces, de infrastructuur, de testproducten en de bevindingen wordt geregeld. Dit kan zowel door centraal standaarden voor beheer op te stellen als door het centraal beheren van bepaalde zaken. Beide mogelijkheden hebben tot doel te voorkomen dat het wiel in de afzonderlijke testsoorten nogmaals uitgevonden wordt.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Definiëren testprocesbeheer;
2. Definiëren infrastructuurbeheer;
3. Definiëren testproductbeheer;
4. Definiëren bevindingenprocedure.

Op mastertestplanniveau kunnen voor de afzonderlijke testsoorten normen en standaarden worden opgesteld, ondersteund door procedures, sjablonen en tools (testwarebeheertools, plannings- en voortgangsbewakingstools). Omdat dit niet wezenlijk anders is dan wanneer dit op testsoortniveau wordt uitgevoerd, wordt voor een beschrijving van deze deelactiviteiten verwezen naar paragraaf 6.2.12 “Inrichten beheer”. In deze paragraaf wordt enkel ingegaan op het specifieke van het inrichten van overkoepelend beheer.

Naast de efficiëntievoordelen dat de beheerprocedures maar één keer hoeven te worden ingericht en beheertools gedeeld kunnen worden tussen de testsoorten, heeft overkoepelend beheer nog een ander groot voordeel: uniformiteit van beheer leidt tot betere beheersing van de afzonderlijke testsoorten door de testmanager, zeker bij uitbesteding of bij testen door externe partijen. In paragraaf 16.3 “Rollen niet als functie beschreven” zijn hiervoor de volgende beheerder-rollen onderkend: testprojectadministrateur, testwarebeheerder en bevindingenbeheerder.

Verder is goed beheer noodzakelijk wanneer de testmanager een verantwoordelijkheid heeft voor de traceerbaarheid van alle tests: kan aangetoond worden dat alles wat getest moet worden ook daadwerkelijk getest is?

1. Definiëren testprocesbeheer

Testprocesbeheer richt zich op het beheren van het testproces in termen van voortgang en kwaliteit, en het inzicht geven in de kwaliteit van het testobject. Hiertoe moet identificatie, registratie, administratie, opslag en interpretatie van de volgende gegevens plaatsvinden;

- voortgang en de besteding van budget en tijd;
- kwaliteitsindicatoren;
- teststatistieken.

Deze informatie vormt de basis voor sturing en rapportage door de testmanager. De testmanager van elke testsoort is verantwoordelijk voor het beheer van het eigen testproces. De testmanager op overkoepelend niveau houdt in de gaten dat dit goed gebeurt. Hiertoe voert de testmanager zo nodig steekproefsgewijs controles op de afzonderlijke testsoorten uit (of laat deze uitvoeren), zowel op de voortgang als op de kwaliteit van het testproces. Dit laatste kan bijvoorbeeld door de methodische testspecialist de traceerbaarheid van testbasis naar testscripts naar testresultaten te laten controleren. Ook kan getoetst worden of voorschriften, aanpak en procedures worden nageleefd. Desgewenst kan hier apart over gerapporteerd worden.

De voortgangsgegevens worden verkregen uit de gegevens en rapportages van de individuele testsoorten en door de gegevens over de overkoepelende activiteiten en producten te registreren. Het bijhouden van dezelfde soorten gegevens voor alle testsoorten is daarom essentieel voor het kunnen opbouwen van overkoepelende statistieken.

2. Definiëren infrastructuurbeheer

Vaak is het zinvol het infrastructuurbeheer centraal over de testsoorten heen te organiseren.

Naast de in paragraaf 6.2.12 beschreven algemene beheertaken zoals backup en recovery, beschikbaarstelling, versiebeheer en onderhoud, zijn de volgende taken specifiek voor centraal beheer:

- het beheren van een tussen testsoorten gedeelde omgeving en/of testtools;
- het afstemmen van de afzonderlijke infrastructuurplanningen;
- het fungeren als intermediair tussen leveranciers (afdeling systeembeheer of de centrale testorganisatie) en afnemers (testsoorten) van de infrastructuur. Centraal beheer kan bijvoorbeeld de opleverprocedure opstellen.

3. Definiëren testproductbeheer

Op mastertestplan niveau worden voor het beheer van de testproducten van de afzonderlijke testsoorten normen en

standaarden opgesteld, ondersteund door procedures, sjablonen en tools. Dit bevordert de herbruikbaarheid van de producten en de communicatie erover. Soms wordt er ook voor gekozen dit beheer centraal uit te voeren. Het is raadzaam aan te sluiten op de algemeen binnen het systeemontwikkel proces geldende normen en standaarden voor documentatie en configuratiebeheer.

4. Definiëren bevindingenprocedure

Omdat een bevindingenprocedure voor het gehele project geldt en niet voor een afzonderlijke testsoort, kan deze procedure het beste op mastertestplanniveau worden gedefinieerd in plaats van per afzonderlijke testsoort. Dit maakt het ook mogelijk om testsoort-overstijgende trends te signaleren. Een beschrijving van de bevindingenprocedure is opgenomen in hoofdstuk 12 “Bevindingenbeheer”.

Producten

Een beschrijving van de bovengenoemde beheerprocessen, vastgelegd in het mastertestplan.

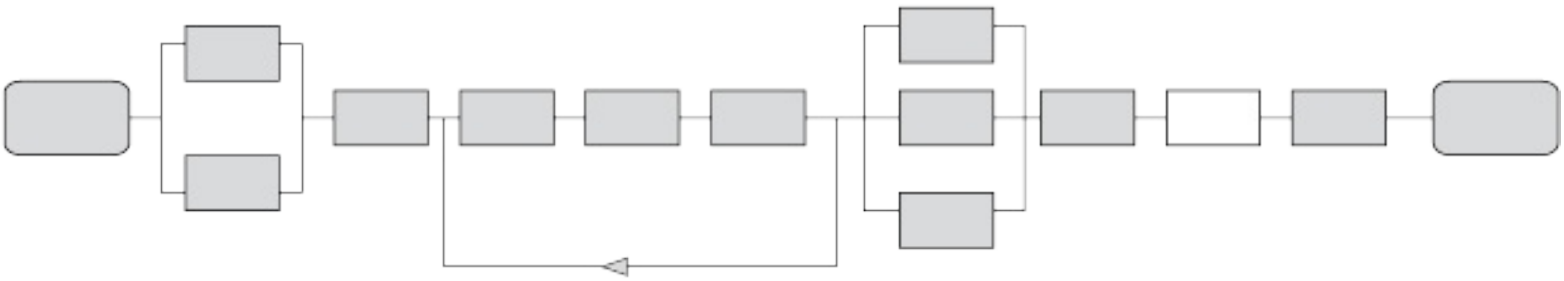
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

5.2.11 Bepalen testprocesrisico’s (& maatregelen)



Doel

Het expliciet benoemen van de risico’s voor het totale testproces waardoor opdrachtgever en andere betrokkenen een beter begrip krijgen van de risico’s voor het testproces en daar in de sturing van het totale voortbrengingsproces rekening mee kunnen houden.

Werkwijze

Bij het uitvoeren van voorgaande activiteiten heeft de testmanager een beeld gekregen van de (on)mogelijkheden voor het testproces, maar ook van bedreigingen en risico’s. In het mastertestplan wordt per risico aangegeven of en zo ja, welke maatregelen zijn genomen ter afdekking of vermindering van het gesignaleerde risico. Denk hierbij aan preventieve maatregelen om risico’s te vermijden, maar eventueel ook aan detectieve maatregelen om problemen tijdig te kunnen signaleren of correctieve maatregelen om de gevolgen te verhelpen.

Uitgediept

De risico’s kunnen onder meer betrekking hebben op:

- *Niet alle relevante partijen betrokken*
  De PRA en daarmee de teststrategie zijn mogelijk onvolledig omdat de input van deze partijen heeft ontbroken. Ook geeft dit het risico dat in een later stadium alsnog grote wijzigingen nodig zijn in het systeem.
- *Voortdurende discussies over de scope van het systeem*
  Wanneer de partijen het niet eens kunnen worden over de scope van het systeem, wordt het moeilijk een betrouwbare strategie, planning en begroting voor testen af te geven en werkt dit zeer verstorend voor het testproces.

Producten

Een beschrijving van de gesignaleerde (test)procesrisico's en mogelijke maatregelen, vastgelegd in het mastertestplan.

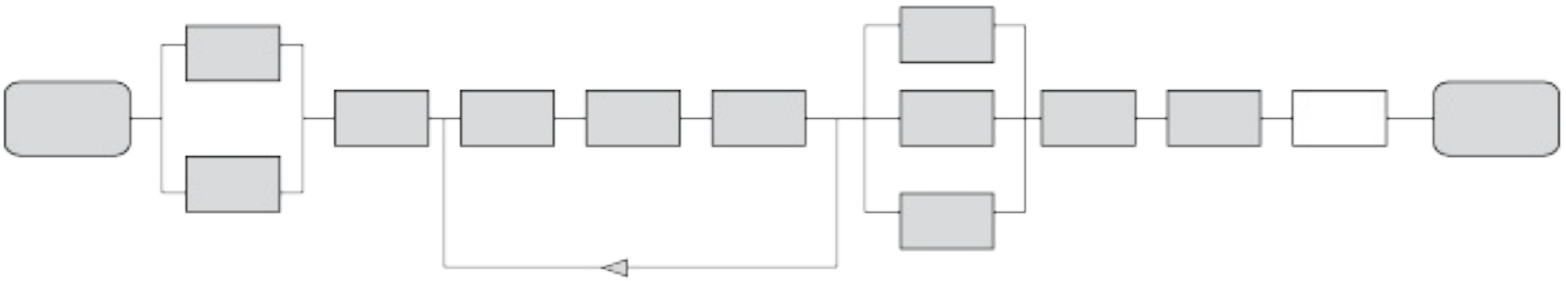
Technieken

Checklist "Risico's testproces" (www.tmap.net).

Tools

Niet van toepassing.

5.2.12 Terugkoppelen en fixeren plan



Doel

Het enerzijds vastleggen van de resultaten van alle tot nu toe uitgevoerde activiteiten en anderzijds het verkrijgen van goedkeuring voor de gekozen aanpak door de opdrachtgever.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Opstellen mastertestplan;
2. Terugkoppeling mastertestplan;
3. Fixeren mastertestplan.

1. Opstellen mastertestplan

De resultaten van alle tot nu toe uitgevoerde activiteiten worden vastgelegd in het mastertestplan. Het mastertestplan bevat in ieder geval de volgende hoofdstukken:

- Opdrachtformulering;
- Teststrategie, inclusief Productrisicoanalyse;
- Aanpak (inclusief doel, korte beschrijving en exitcriteria per testsoort);
- Organisatie;
- Infrastructuur;

- Beheer;
- Bedreigingen, risico's en maatregelen;
- Globale begroting en planning.

2. Terugkoppeling mastertestplan

Het mastertestplan met de resultaten van voorgaande activiteiten wordt teruggekoppeld met de opdrachtgever en andere betrokken partijen voor goedkeuring of bijstelling. Dit maakt de te volgen testaanpak transparant en stuurbaar, geheel in lijn met BDTM.

Hoewel van alles kan veranderen, zien we in de praktijk dat vooral de combinatie van gekozen teststrategie, begroting en planning wordt bijgesteld. Door de strategie aan te passen (waarbij de risicoanalyse in principe niet verandert), kan de testmanager de opdrachtgever laten sturen op de afweging tussen testinspanning versus testzwaarte en de risico's die daarmee gelopen worden. Dit resulteert in een aangepaste teststrategie, waarbij het schrappen van testzwaarte wordt getoond door ○ in plaats van ● (en extra testzwaarte door ● in plaats van ●), bijvoorbeeld:

Kenmerk	PRA-RK	Toetsen	Ontwikkeltest	ST	GAT	PAT
Functionaliteit						
- deelsys 1	A	●	● ○	● ● ●	● ●	
…						

3. Fixeren mastertestplan

Na voorgaande activiteit dient de testmanager het mastertestplan ter goedkeuring voor te leggen aan de opdrachtgever en andere betrokkenen.

Het plan wordt nu als formeel testproduct onder configuratiebeheer geplaatst. Daarnaast kan een presentatie, bijvoorbeeld aan de stuurgroep en de diverse betrokken partijen, bijdragen aan het verkrijgen van goedkeuring én, minstens zo belangrijk, draagvlak binnen de organisatie.

Producten

Het mastertestplan.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

5.3 Fase Beheer van het totale testproces

Doel

Het geven aan de opdrachtgever van voldoende inzicht in én sturingsmogelijkheden over:

- de voortgang van het testproces;
- de kwaliteit en risico's van het testobject;
- de kwaliteit van het testproces.

Hiervoor dient de testmanager het totale testproces op optimale wijze te beheersen en hierover te rapporteren.

Context

Deze activiteit heeft betrekking op alle bij het mastertestplan betrokken test- en toetssoorten, zoals al is aangegeven in paragraaf 5.2 [Context]. Het testen kan betrekking hebben op nieuwbouw, onderhoud, migratie, een pakketimplementatie of een mix hiervan. De ontwikkelaanpak kan waterval, iteratief, agile of ook een mix zijn. Afhankelijk van de gemaakte afspraken kan de testmanager ten opzichte van een testsoort een sturende rol, een coördinerende rol of enkel een waarnemende rol hebben.

Randvoorwaarden

Deze activiteit start na het opstellen van het mastertestplan of wanneer één of meer van de betrokken test- en toetssoorten gestart zijn.

Werkwijze

De testmanager en de beheerder(s) voeren de hun in het mastertestplan toegekende activiteiten uit. Zij beheren het testproces, de infrastructuur en de testproducten, mede op basis van de door de testsoorten aangeleverde gegevens.

Op basis van deze gegevens analyseert de testmanager mogelijke trends. Tevens zorgt de testmanager dat hij bijzonder goed op de hoogte blijft van de ontwikkelingen buiten het testen, zoals vertraging bij de ontwikkelaars, komende grote wijzigingsvoorstellen en projectbijsturing. Indien nodig stelt de testmanager de opdrachtgever bepaalde sturingsmaatregelen voor.

Informatie is het belangrijkste product van testen. Hiertoe verzorgt de testmanager verschillende soorten rapportages aan de diverse doelgroepen, rekening houdend met de BDTM elementen Resultaat, Risico's, Tijd en Kosten.

Rollen/verantwoordelijkheden

De primair verantwoordelijke rol voor (de coördinatie van) het beheer van het totale testproces is de testmanager of overall testcoördinator.

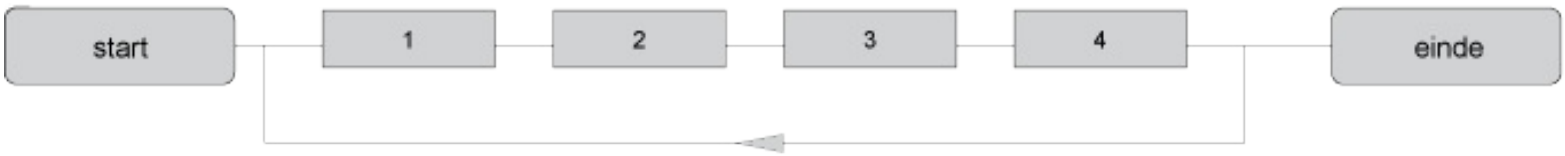
Daarnaast kunnen op overkoepelend testniveau bepaalde rollen zijn ingericht voor beheer en ondersteuning, zowel voor het overkoepelende niveau als voor de afzonderlijke testsoorten.

Activiteiten

Het beheer van het totale testproces omvat de volgende activiteiten:

1. Uitvoeren beheer;
2. Bewaken;
3. Rapporteren;
4. Bijsturen.

Onderstaand schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten, waarbij de pijl duidelijk maakt dat de activiteiten nadrukkelijk iteratief verlopen:



Figuur 5.9: Beheer van het totale testproces.

5.3.1 Uitvoeren beheer

Doel

Het op overkoepelend niveau beheren van het testproces, de infrastructuur en de testproducten om voortdurend inzicht te kunnen bieden in de voortgang en kwaliteit van het totale testproces en de kwaliteit van het testobject.

Werkwijze

De beheer-activiteit kan in twee subactiviteiten worden onderscheiden:

- Conform de in het mastertestplan vastgestelde procedures worden de volgende onderscheiden vormen van beheer uitgevoerd: het beheer van het testproces, infrastructuurbeheer, het testproductbeheer en het bevindingenbeheer (zie paragraaf 5.2.10 “Inrichten beheer”).
- De testsoorten worden ondersteund met én gecontroleerd op het toepassen van de overkoepelende normen en standaarden voor beheer.

Deze werkzaamheden vallen onder de rol van beheerder of testmanager, zoals gedefinieerd in paragraaf 5.2.8 “Definiëren organisatie”. In paragraaf 16.3 “Rollen niet als functie beschreven” zijn de volgende beheerder-rollen onderkend: testprojectadministrateur, testwarebeheerder en bevindingenbeheerder.

Producten

Een beheerd testproces.

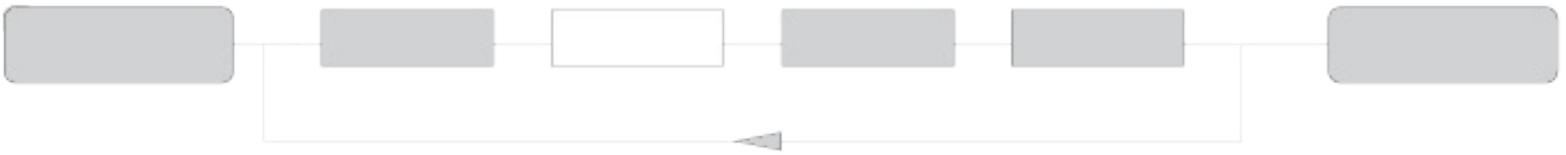
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

5.3.2 Bewaken



Doel

Het op basis van interne beheerde gegevens en externe informatie bewaken van het totale testproces

Werkwijze

De voornaamste en moeilijkste taak van de testmanager is de bewaking van de uitvoering van het mastertestplan.

Hoewel dit in onderstaande paragraaf met name als een instrumentele activiteit is beschreven, is dit met name een *communicatieve activiteit* met *medewerker-, stakeholder- en verwachtingsmanagement* waarvoor een grote dosis sociale en communicatieve vaardigheden nodig zijn.

Algemeen gesteld moeten de in het plan beschreven testsoorten volgens een bepaald tijdpad en in een bepaalde volgorde uitgevoerd worden. De testmanager moet de vraag kunnen beantwoorden of het verantwoord is om te starten met testen. Tegen het einde van de testuitvoering moet de testmanager de vraag kunnen beantwoorden of het verantwoord is om te stoppen.

De praktijk leert dat afwijkingen van het oorspronkelijke plan gaan optreden. Deze afwijkingen kunnen een oorzaak hebben die zowel binnen als buiten het testproces kan liggen. Voorbeelden van het eerste zijn dat het specificeren of uitvoeren van testgevallen langzamer gaat dan verwacht (de testers zijn minder productief dan verwacht) of dat bepaalde testsoorten niet volgens planning kunnen starten met testen (door een verkeerde inschatting vooraf). Voorbeelden van het tweede zijn: onvoldoende kwaliteit van de testbasis of het testobject, extra wijzigingsvoorstellen of uitloop van ontwerp of bouw.

Voor de testmanager is het zaak om deze gebeurtenissen of trends zo vroeg mogelijk te signaleren. Maatregelen om een negatieve trend bij te sturen kunnen dan tijdig worden genomen. Dit is vrijwel altijd beter, goedkoper en sneller.

De informatie voor de gebeurtenissen of trends komt van een aantal kanten:

- intern beheerde gegevens op overkoepelend niveau;
- gegevens aangeleverd door (rapportages van) de testsoorten;
- informatie van buiten het testproces, afkomstig uit documentatie als notulen of memo's, maar zeker ook uit mondeling overleg als projectoverleg, bilaterale overleggen, stand-up meetings, enzovoort.

Met name de mondelinge bron van informatie benadrukt het belang van goede (sociale) vaardigheden en externe gerichtheid voor de testmanager.

Op basis van deze gegevens analyseert de testmanager mogelijke trends en probeert hij bedreigingen (of juist kansen) tijdig te signaleren. Hiertoe voert de testmanager de volgende stappen uit:

1. Analyseren gebeurtenis, inschatten risico's en definiëren maatregelen
2. Afstemming met opdrachtgever en andere belanghebbenden (optioneel, afhankelijk van tolerantie)

Deze stappen zijn uitgebreid beschreven in paragraaf 6.3.2 "Bewaken". In deze verdere paragraaf wordt volstaan met enkele specifieke zaken voor het mastertestplan toe te lichten.

Uitgediept

Onderstaand wat voorbeelden van veel voorkomende gebeurtenissen met oorzaken, gevolgen en maatregelen die de testmanager kan voorstellen:

Gebeurtenis	***Een testsoort loopt uit de planning***
Mogelijke oorzaken	De oorzaken hiervoor kunnen in het traject liggen vóór het testtraject. Denk aan tegenvallende kwaliteit testbasis/-object, te late oplevering door ontwerp/bouw, testinfrastructuur niet op tijd geïnstalleerd, installatie testobject in testomgeving loopt (sterk) uit. Ook kan de oorzaak bij de testsoort zelf liggen. Denk aan onvoldoende resources, verkeerde planning of onvoldoende bewaking van de voortgang.
Gevolgen voor het testproces	Opvolgende testsoorten kunnen later starten
Mogelijke maatregelen	• Bij iteratieve ontwikkeling in overleg met ontwikkelaars: systeemfunctionaliteit beperken • Inzet van extra testcapaciteit • Opvolgende testsoorten starten parallel aan de lopende testsoort, met inefficiëntie tot gevolg (twee testsoorten doen dezelfde bevindingen) • Er wordt besloten om bepaalde testsoorten of testvormen in elkaar te schuiven waardoor ze gezamenlijk uitgevoerd worden. • Aanvullend budget vragen • Overkoepelende teststrategie herzien, extra risico nemen • Projectplanning herzien • …

Gebeurtenis	***Een testsoort heeft geen tijd/middelen meer om alle benodigde (her)tests uit te voeren***
Mogelijke	Ook hier kunnen de oorzaken buiten het testen liggen, zie boven. Maar ook interne oorzaken zijn mogelijk: verkeerde

oorzaken	planning en begroting, onverwacht wegvallende resources, onvoldoende productiviteit.
Gevolgen voor het testproces	• Opvolgende testsoorten krijgen een testobject van onvoldoende of onduidelijke kwaliteit binnen • Opvolgende testsoorten lopen uit óf moeten stoppen na onvoldoende testen • Boeggolfwerking: het project lijkt op volle snelheid conform planning door te stomen, maar er ontstaat een steeds grotere boeggolf van achterstallig werk/onvoldoende kwaliteit. Pas tegen het einde wordt pas op de plaats gemaakt, waarna de enorme hoeveelheid herstelwerk en uitloop duidelijk wordt.
Mogelijke maatregelen	• Inzet van extra testcapaciteit • Uitloop toestaan van de betreffende testsoort, zie boven • Aanvullend budget vragen • Overkoepelende teststrategie herzien, extra risico nemen • Projectplanning herzien • …

Gebeurtenis	***Onduidelijkheid over de scope van de testsoorten***
Mogelijke oorzaken	• Gebrek aan informatie bij opstellen overkoepelende teststrategie • Onvoldoende afstemming bij opstellen overkoepelende teststrategie • Voortschrijdend inzicht of gewijzigde functionaliteit/scope project maakt nieuwe te testen aspecten duidelijk
Gevolgen voor het testproces	• Bepaalde aspecten worden niet getest óf • Extra tijd/geld/middelen nodig om aspect alsnog te testen • Uitloop op planning
Mogelijke maatregelen	• Opnieuw afstemmen over scope testsoorten, overkoepelende teststrategie herzien • Inzet van extra testcapaciteit bij betreffende testsoorten • Uitloop toestaan van de betreffende testsoorten, zie boven • Aanvullend budget vragen • Projectplanning herzien • …

Gebeurtenis	***Relatief grote aantallen bevindingen op bepaalde kenmerken of deelobjecten in bepaalde testsoorten***
Mogelijke oorzaken	Afhankelijk van de oorzaak: onvoldoende kwaliteit van testobject of testbasis
Gevolgen voor het testproces	• Extra tijd/geld/middelen nodig voor hertesten én voor aanvullende tests • Uitloop op planning
Mogelijke maatregelen	• Opnieuw afstemmen, overkoepelende teststrategie herzien • Adviseren dat voorliggende test- en toetsactiviteiten grondiger worden uitgevoerd voor betreffende kenmerken/deelobjecten • Inzet van extra testcapaciteit bij betreffende testsoort • Uitloop toestaan van de betreffende testsoort, zie boven • Aanvullende tests inplannen bij volgende testsoort • Aanvullend budget vragen • Projectplanning herzien • …

NB. De bovenstaande tabellen dienen als voorbeeld. Ze zijn bedoeld om handvatten te geven en niet om een volledig beeld te geven.

Producten

Voorgestelde sturingsmaatregelen.

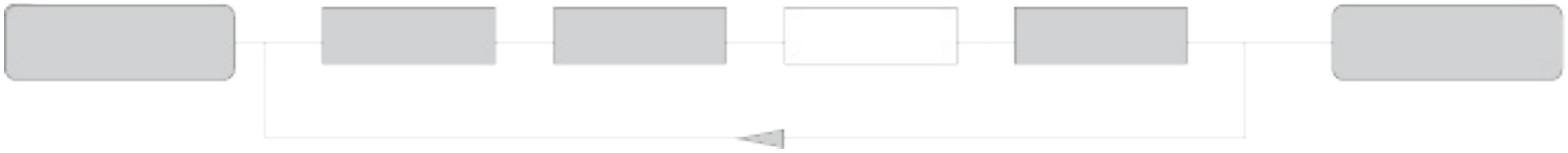
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

5.3.3 Rapporteren



Doel

Het opstellen van rapportages om inzicht te verschaffen in zowel de kwaliteit van het testobject als de voortgang en kwaliteit van de afzonderlijke testsoorten en het totale testproces. Deze rapportages zorgen ervoor dat de opdrachtgever en andere belanghebbenden effectief kunnen sturen op het verloop van het testtraject.

Werkwijze

Gedurende het testproject stelt de testmanager verschillende rapportages op. In het mastertestplan zijn in het hoofdstuk “Beheer” de vorm en de frequentie van rapportage vastgesteld. Periodiek wordt gerapporteerd over de kwaliteit van het testobject en de voortgang en kwaliteit van het testproces. Naast de periodieke rapportages, kan door de opdrachtgever of andere betrokkenen worden gevraagd om ad-hoc rapportages. Het bekendste voorbeeld hiervan is het risicorapport, om de mogelijke gevolgen van een bedreiging of risico voor het testtraject in kaart te brengen en maatregelen voor te stellen. Daarnaast kan ook opeens gevraagd worden om een extra voortgangsrapport, bijvoorbeeld als meest actuele input voor een stuurgroep- of projectmanagementoverleg. Aan het einde van het testproces worden een vrijgaveadvies en eindrapport gemaakt.

Met al deze informatie krijgen opdrachtgever, projectmanager en andere betrokkenen inzicht in de mate waarin:

- het beoogde resultaat is behaald;
- de risico’s bij in productie name bekend en zo klein mogelijk zijn binnen de gestelde randvoorwaarden;
- dit binnen het vastgestelde budget en de vastgestelde (doorloop)tijd gebeurt.

Ofwel, dit zijn de BDTM aspecten Resultaat, Risico’s, Tijd en Kosten. Het geven van inzicht betekent dat het rapport moet aansluiten op de belevingswereld van de ontvanger(s) ervan.

De rapportages zijn gebaseerd op de gegevens zoals vastgelegd volgens paragraaf 5.2.10 “Inrichten beheer”. Deze gegevens zijn grotendeels afkomstig van de testsoorten.

> **Tip**
>
> Bij rapportages komt het belang van tools om de hoek kijken. Het helpt bij de standaard rapportages, maar vooral ook bij ad-hoc rapportages, wanneer snelheid geboden is. De testmanager moet daarvoor goed met de betreffende tools (zie onder) om kunnen omgaan!

De belangrijkste rapportages zijn:

- voortgangsrapport;
- risicorapport;
- vrijgaveadvies;
- eindrapport.

Deze rapportages staan uitgebreid beschreven in paragraaf 6.3.3 “Rapporteren”.

Op de volgende pagina staat een voorbeeld van (een deel van) een voortgangsrapport waarin Resultaat en Tijd in een spreadsheet gecombineerd zijn.

Korte omschrijving project: Internetsite met applicaties waarmee klanten zelf mutaties kunnen doen																	
Test infrastrucutuur		Development	beschikbaar, geen echte testdata														
		Systeem	beschikbaar vanaf 01-03-2006.														
		Acceptatie	De acceptatieomgeving moet nog worden ingericht. De eerste oplevering vindt plaats op 01-12-2006. Tot die tijd test de business mee in de systeemtestomgeving														
				Systeemtest							Acceptatietest + Productieacceptatietest						
Risico / aandachtsgebied	Testobject - ID	Testeenheid	Testcases beschikbaar ? J / N	Start ST	Eind ST	Getest?	Resultaten OK/ NOK	Openstaande bevindingen	Totaal aantal	Opmerkingen	Start GAT / PAT	Eind GAT / PAT	Getest ?	Resultaten OK/ NOK	Openstaande bevindingen	Totaal aantal	Opmerkingen
Functionaliteit																	
	Internet front-end		j	21/05/06	25/05/06	25%	n	10	1		3/06/06	7/06/06	0%	n			
	GUI01	Navigatie	j	21/05/06	25/05/06	25%	n				3/06/06	7/06/06	0%	n			
	GUI02	Lay-out	j	21/05/06	25/05/06	25%	n				3/06/06	7/06/06	0%	n			
	[application] - AB		j	11/05/06	25/05/06	80%	n	2.8	2		29/05/06	6/10/06	0%	n			
	ABa01	DS1	j	11/05/06	25/05/06	100%	j				29/05/06	6/10/06	0%	n			
	ABa02	DS2	j	11/05/06	25/05/06	60%	n	2			29/05/06	6/10/06	0%	n			
	ABr02	DS3	j	11/05/06	25/05/06	60%	n	8			29/05/06	6/10/06	0%	n			
	Content Management System		j	01/05/06	15/05/06	100%	j				18/05/06	28/05/06	50%	n	4.5	2	
	CM001	DS4	j	01/05/06	15/05/06	100%	j				18/05/06	28/05/06	50%	n			
	CM002	DS5	j	01/05/06	15/05/06	80%	n			Niet alle testen uitgevoerd, toch naar acceptatie	18/05/06	28/05/06	50%	n	4.5		
	Mangement Informatie		j	12/05/06	15/05/06	100%	j				19/05/06	1/06/06	0%	n	6	1	
	MI001	Rapportage X	j	12/05/06	15/05/06	100%	j						0%	n			Er is besloten door het projectmanagement wegens tijdgebrek dit onderdeel niet mee te nemen in de GAT. In ST is dit onderdeel goedgekeurd.
	MI002	Rapportage Y	j	12/05/06	15/05/06	100%	j				19/05/06	1/06/06	0%	n	6		
Inpasbaarheid																	
	Afdelingen		n/a				n/a				3/06/06	10/06/06	0%	n			
	Afd001	Bedrijfsproces 1	n/a				n/a				3/06/06	10/06/06	0%	n			
	Afd002	Bedrijfsproces 2	n/a				n/a				3/06/06	10/06/06	0%	n			
	Afd003	Helpdeskprocedure	n/a				n/a				3/06/06	10/06/06	0%	n			
Bruikbaarheid																	
	Usabilitytest		n	05/05/06	10/05/06	100%	j						0%	n/a			
	US001	Extern usabilitylab	n	05/05/06	10/05/06	100%	j			Is door een extern bureau op basis van het prototype uitgevoerd en goed bevonden.			0%	n/a			
	US002	Extern usabilitylab	n	05/05/06	10/05/06	0%	n			Besloten door projectmanagement om deze test niet extern te laten uitvoeren in verband met budgetproblemen, i.p.v. daarvan wordt er extra aan besteed in de interne opleiding.			0%	n/a			
Beveiliging																	
	Beveiligingstesten		n/a				n/a				3/06/06	7/06/06	0%	n			
	BE001	Hackerstest (exern)	n/a				n/a				15/06/06	25/06/06	0%	n			
	BE002	Autorisaties	n/a				n/a							n			
	BE003	[application] - SSL	n/a				n/a				3/06/06	7/06/06	0%	n			
	BE004	[application]: transacties - autorisatieniveau 3	n/a				n/a				3/06/06	7/06/06	0%	n			
Beheerbaarheid																	
	Webserver		n/a				n/a				18/05/06	7/06/06	0%	n	9	1	Er zijn bevindingen geconstateerd tijdens de PAT, maar niet alle medewerkers registreren deze. Het beeld is daarom onvolledig.
	WS001	Procedure backup / restore database [application]	n/a				n/a				18/05/06	28/05/06	60%	n	9		
	WS002	Checklist Beheerbaarheid	n/a				n/a				3/06/06	7/06/06	0%	n			
	WS003	Uitwijkprocedure	n/a				n/a				3/06/06	7/06/06	0%	n			
Performance																	
	Performancetest		n/a				n/a				3/06/06	7/06/06	0%	n			
	PE001	Loadtest	n/a				n/a				3/06/06	7/06/06	0%	n			
	PE002	Stresstest	n/a				n/a				3/06/06	7/06/06	0%	n			

Producten

Rapportages (voortgangsrapport, risicorapport, vrijgaveadvies, eindrapport).
Ervaringsgegevens.
Kosten/baten analyse.

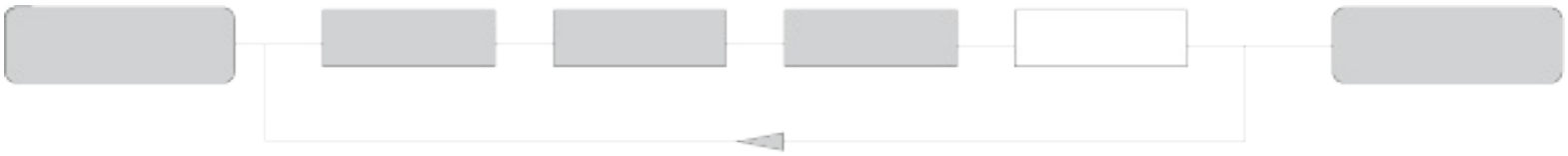
Technieken

Checklist “Evaluatie testproces” (www.tmap.net).
Metrics (hoofdstuk 13).

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

5.3.4 Bijsturen



Doel

Het (indien nodig in overleg met opdrachtgever) bijsturen van het totale testproces

Werkwijze

Wanneer de in paragraaf 5.3.2 “Bewaken” voorgestelde maatregelen zijn gerapporteerd en de opdrachtgever akkoord is en een keuze heeft gemaakt uit één of meer van de mogelijke alternatieven, kan de testmanager deze effectueren. Hiertoe voert hij de volgende stappen uit:

1. Doorvoeren maatregelen en evalueren effectiviteit;
2. Aanpassen producten uit de planfase (optioneel, afhankelijk van tolerantie);
3. Terugkoppelen aan opdrachtgever.

Uitgediept

Bijstelling strategie

In veel gevallen is een bijstelling van de teststrategie nodig. Een goede reden om de strategie te wijzigen is dat bepaalde tests veel meer of juist minder bevindingen opleveren dan verwacht, of dat tijdens het testtraject blijkt dat de productrisico’s niet juist zijn ingeschat of dat er aanvullende productrisico’s worden onderkend. Er kan dan besloten worden om extra tests te specificeren en uit te voeren of om geplande tests maar gedeeltelijk uit te voeren of zelfs te laten vervallen. Een minder goede reden om de strategie te wijzigen is wanneer men uitloop van de rest van het ontwikkelproces in kosten of tijd wil compenseren door minder te testen.

Met de strategiebepaling als basis overlegt de testmanager met de opdrachtgever over de veranderde risicoinschatting en over wat meer of minder getest moet worden. Door aan te geven wat minder getest gaat worden in relatie tot de onderkende productrisico’s, kan de testmanager op goede wijze rapporteren over het hierdoor eventueel ontstane hogere risico dat overblijft na de test.

Onderstaande tabel is een voorbeeld van een gewijzigde PRA en teststrategie op het overkoepelende niveau.

Voorbeeld:

Kenmerk	RK	Toetsen	Ontwikkeltest	ST	GAT	PAT
Functionaliteit						
- deelsys 1	A	●	●	● ● ○	● ○	
- deelsys 2	A (B)					

		●	●	● ●		
- totaal	B (C)			● ●	●	
Gebruiks-vriendelijkheid	B	●			● ○	
Performance						
- online	B			●		●
- batch	C					●
Beveiliging	B (A)	●			●	● ● ○
Inpasbaarheid	B	●			● ● ○	

In dit voorbeeld is de toegekende risicoklasse voor functionaliteit hoger geworden voor deelsysteem 2 en het totale systeem. De testzwaarte in de systeemtest en gebruikersacceptatietest is voor deelsysteem 1 juist lichter geworden (respectievelijk van 3 naar 2 en van 2 naar 1, gesymboliseerd door ○). Wel wordt de systeemtest op het integrale systeem zwaarder gemaakt, van 1 naar 2, gesymboliseerd door ●. Duidelijk mag zijn dat het risico groter wordt en de testzwaarte minder. Dit vraagt vanzelfsprekend een toelichting én een goede risico-inschatting van de testmanager voor de opdrachtgever (valt buiten het bestek van dit voorbeeld).

De meest voorkomende en te verwachten wijzigingen hebben betrekking op de planning. Indien de herplanning van één testsoort gevolgen heeft voor de planning van andere testsoorten en projectonderdelen, worden alle betrokken partijen geïnformeerd en waar nodig dient hun instemming te worden verkregen.

Producten

Sturingsmaatregelen.
Bijgesteld plan, in de vorm van een nieuwe versie of als aanvulling op het originele plan.

Technieken

Niet van toepassing.

Tools

Plannings- en voortgangsbewakingstool.
Workflowtool.

5.4 Generieke Test Afspraken

De zogenaamde Generieke Test Afspraken (GTA) vertonen gelijkenis met een mastertestplan en komen daar soms zelfs voor in de plaats. Standaard geldt het mastertestplan voor één project. In veel gevallen, zoals bij outsourcing, offshoring, onderhoud of bij programma's, overstijgt het testproces echter een enkel project. De initiële investering begint meestal pas rendabel te worden bij opvolgende releases van het systeem, dus bij het onderhoud. Om te voorkomen dat bij elke release weer opnieuw afspraken gemaakt moeten worden over het wat en hoe van het testen, worden deze vastgelegd in het GTA-document. Hierin staan de algemene afspraken over bijvoorbeeld het testproces, standaard strategie, de manier van begroten, procedures, organisatie, communicatie, documentatie, enzovoort. Het is hiermee een overkoepelende aanpak hoe testtrajecten geregeld en ingericht worden, geldend voor (alle) toekomstige releases. Dat is dus niet per project / release waarin de afstemming tussen testsoorten van die release beschreven wordt. Op basis van de GTA wordt er daarna gekozen om per release als enige nog een gedetailleerd mastertestplan op te stellen, of om een mastertestplan met afzonderlijke testplannen per testsoort op te stellen of om alleen de afzonderlijke testplannen te maken.

Naast bovengenoemde situaties worden ook voor iteratieve projecten vaak GTA gemaakt. Per increment stelt de testmanager dan een mastertestplan op of worden zelfs rechtstreeks de afzonderlijke testplannen opgesteld.

Een ander voordeel van GTA speelt bij niet-gepland (ad-hoc) onderhoud. Hierbij heeft het oplossen van het productieprobleem de hoogste prioriteit. Hoewel dit weer tot gevolg heeft dat niet alle benodigde stappen van een gestructureerde testaanpak uitgevoerd kunnen worden, zijn juist dan GTA onmisbaar, omdat hierin staat welke testactiviteiten bij ad-hoc onderhoud essentieel zijn en altijd uitgevoerd moeten worden. Als hiernaast ook nog een

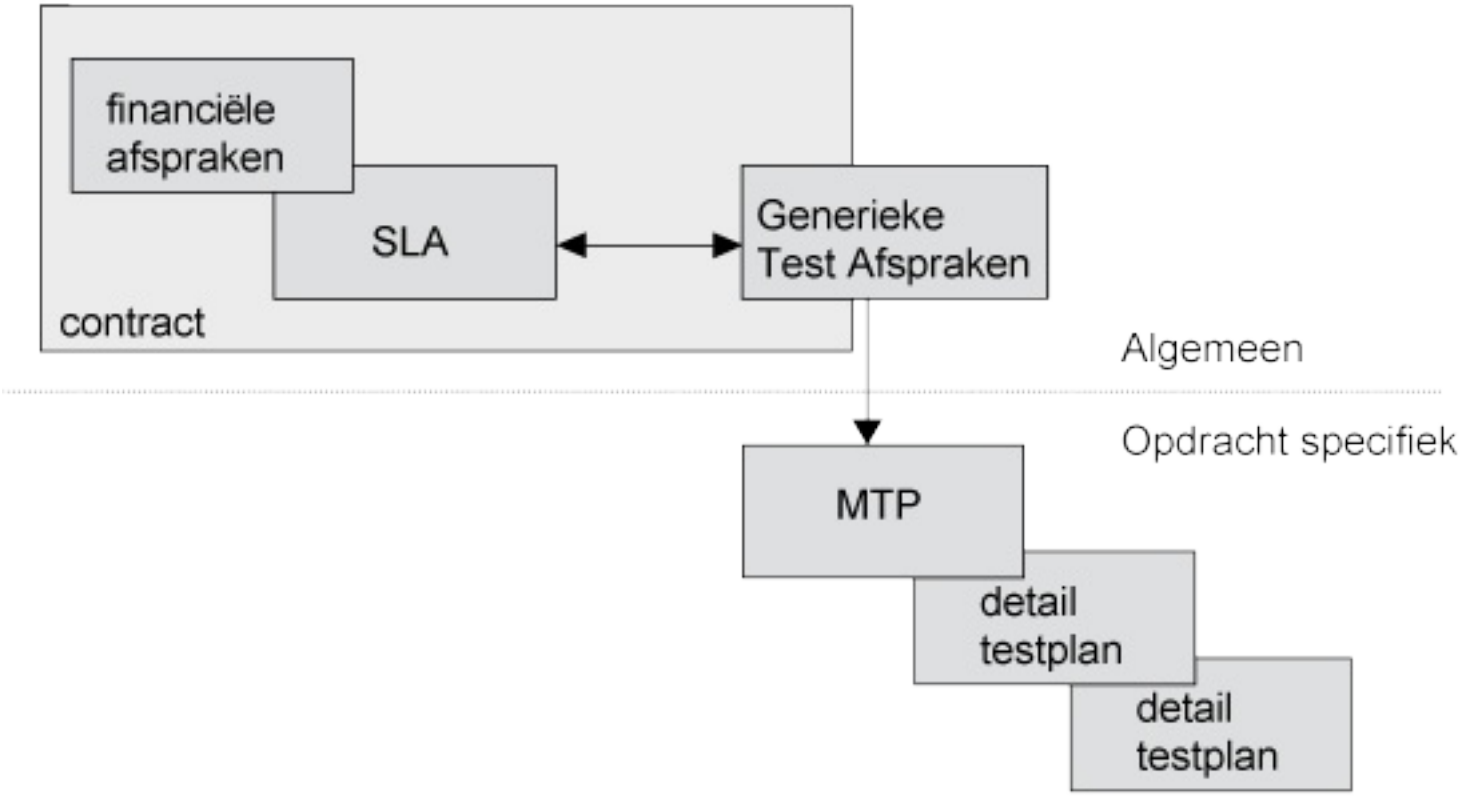
“eenvoudig” op de teststrategie aan te passen schaalbare regressietestset aanwezig is, dan kan “zelfs” bij ad-hoc correctief onderhoud een kwalitatief goede test worden uitgevoerd.

GTA zijn feitelijk een soort Service Level Agreements tussen opdrachtgever en opdrachtnemer. Deze is weer onderdeel van het contract, met daarin ook bijvoorbeeld afspraken over tijdige levering van capaciteit, reactietijden, inwerken van nieuwe mensen, prijsstelling, enzovoort.

Per release of per project vult de testmanager de GTA aan met informatie zoals het invullen van wat getest gaat worden, bemensing en mijlpalen, tot het detailniveau van een mastertestplan. De indeling van een GTA-document en een mastertestplan zijn daarmee gelijk, het verschil zit in de inhoud. Voorbeeld: in de paragraaf Mijlpalen staat in een GTA op welke manier dit aangegeven moet worden, in een mastertestplan staan de feitelijke mijlpalen voor het specifieke project.

Op www.tmap.net is de inhoudsopgave van een GTA-document opgenomen.

Gezien het feit dat de GTA geen planning of mijlpalen bevatten, is het formeel gezien geen plan. Andere namen waaronder GTA bekend staat zijn GMTA (Generieke Master Test Afspraken), MTV (Master Test Visie) en GMTP (Generieke Master Test Plan of Generiek Master Test Protocol).



Figuur 5.10, Relatie contract, GTA, mastertestplan (MTP) en detailtestplan.

Noot

1 Het testen zelf kan er nooit voor zorgen dat de schade bij in-productie-name vermindert. Vanuit de testresultaten worden bevindingen en daarmee potentiële schade aangetoond. Op basis hiervan wordt besloten om al dan niet aanpassingen in het testobject door te voeren die de schade bij in-productie-name wegnemen of verminderen.

6 Acceptatie- en systeemtesten

6.1 Inleiding

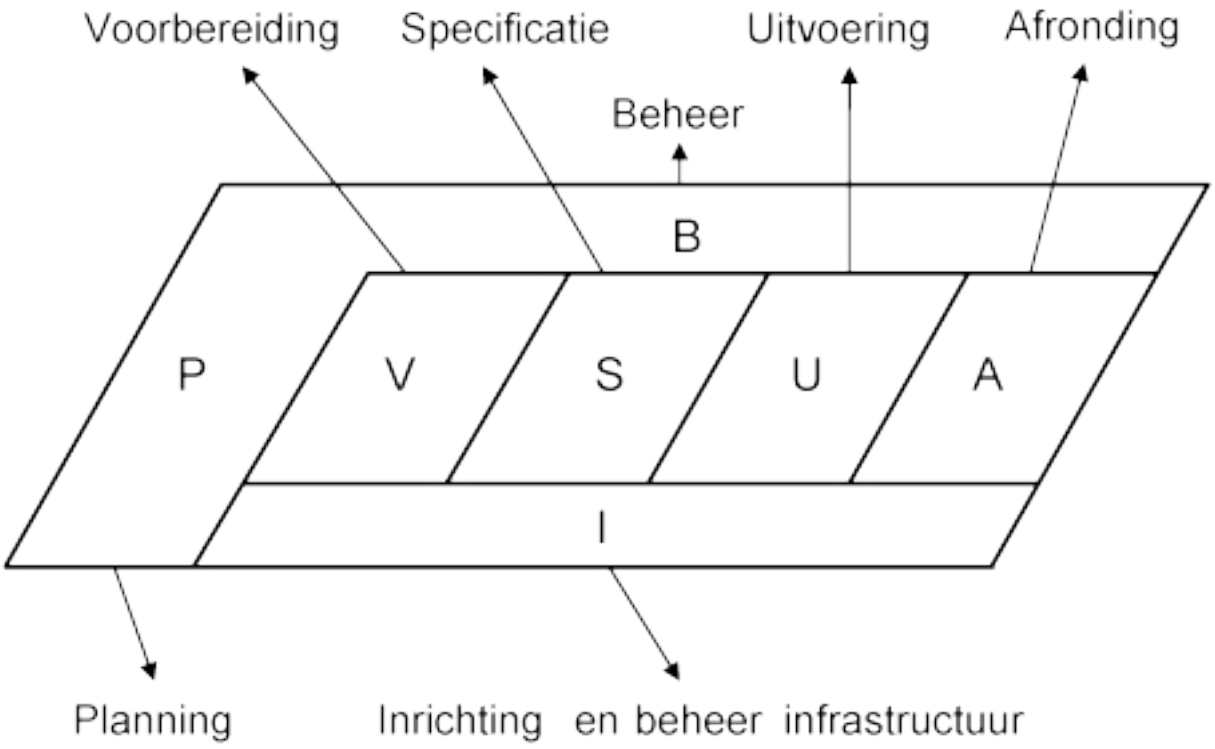
Acceptatietest en systeemtest

In dit hoofdstuk wordt het faseringsmodel van TMap, met bijhorende activiteiten, beschreven voor de testsoorten acceptatietest en systeemtest. Zowel de acceptatietest als de systeemtest zijn feitelijk te beschouwen (en daarmee ook te organiseren) als op zichzelf staande processen. Ze hebben een eigen testplan, een eigen budget en vaak ook een eigen testomgeving waarop de test wordt uitgevoerd. Het zijn processen die parallel lopen aan het ontwikkelproces en bij voorkeur starten tijdens het opstellen van de functionele specificaties.

Zoals ook in hoofdstuk 2 “Kader en belang van testen” is beschreven, kan in een ontwikkelproces een scheiding worden aangebracht in verantwoordelijkheden tussen de opdrachtgever enerzijds en de leverancier anderzijds. In de context van het testen wordt de eerste groep samengevat als de accepterende (vragende) partij en de tweede groep als leverende partij. Ieder van deze partijen heeft zijn eigen verantwoordelijkheid in het testen. De leverancier voert de systeemtest uit om vast te stellen of het systeem aan de functionele en technische specificaties voldoet. Daarmee wordt aangetoond dat wat geleverd moet worden ook werkelijk wordt geleverd. Nadat de leverancier de systeemtest heeft uitgevoerd, de aangetroffen fouten heeft hersteld en met een positief resultaat aan een hertest heeft onderworpen, wordt het systeem ter acceptatie aan de opdrachtgever aangeboden. Deze accepterende partij wil met de test vaststellen of wat gevraagd is ook werkelijk wordt verkregen en of ze met het product kunnen doen wat ze willen/moeten doen.

TMap faseringsmodel

Het proces van de acceptatietest en de systeemtest bestaat uit een aantal verschillende activiteiten. Voor het overzichtelijk in kaart brengen van de verschillende activiteiten, met hun onderlinge volgorde en afhankelijkheden, bestaat het TMap faseringsmodel. Dit is een generiek model en het is toepasbaar voor beide testsoorten. De acceptatietest en de systeemtest geven wel elk hun eigen specifieke invulling aan het faseringsmodel. In het TMap faseringsmodel zijn de testactiviteiten verdeeld over een zevental fasen (zie figuur 6.1 “TMap faseringsmodel”). Dit zijn de fasen Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding.



Figuur 6.1: TMap faseringsmodel.

In de fase Planning (paragraaf 6.2) wordt door de testmanager een samenhangende en door de opdrachtgever gedragen aanpak geformuleerd waarmee de testopdracht goed uitgevoerd kan worden. Dit wordt dan vastgelegd in het testplan.

Tijdens de fase Beheer (paragraaf 6.3) worden de activiteiten uit het testplan uitgevoerd, bewaakt en eventueel bijgestuurd. De fase Inrichting en beheer infrastructuur (paragraaf 6.4) heeft als doel om zorg te dragen voor de benodigde testinfrastructuur die wordt gebruikt bij de verschillende TMap fasen en activiteiten. De fase Voorbereiding (paragraaf 6.5) heeft als doel om te kunnen beschikken over een, met de opdrachtgever van de test overeengekomen, testbasis die voldoende van kwaliteit is voor het ontwerpen van de testgevallen. De tests worden gespecificeerd in de fase Specificatie (paragraaf 6.6) en uitgevoerd in de fase Uitvoering (paragraaf 6.7). Zo wordt inzicht verkregen in de kwaliteit van het testobject. Tijdens de fase Afronding (paragraaf 6.8) wordt de testopdracht afgerond. Er is dan gelegenheid om lessen te trekken uit ervaringen die zijn opgedaan. Ook worden activiteiten uitgevoerd om hergebruik van producten te garanderen.

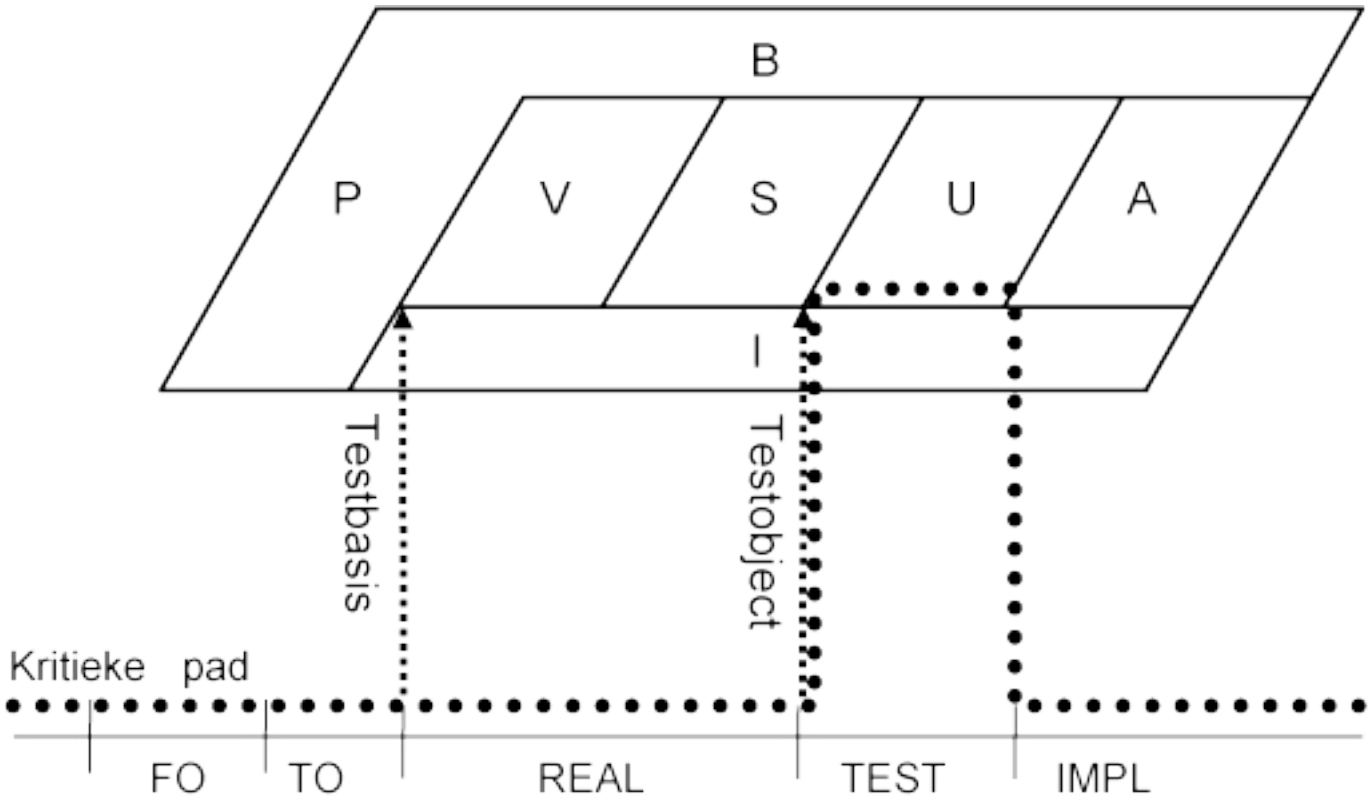
De hiervoor beschreven fasen hoeven niet altijd geheel sequentieel te worden uitgevoerd. Het is bijvoorbeeld heel goed mogelijk dat testgevallen voor een deel van de test nog gespecificeerd worden (fase Specificatie), terwijl de testuitvoering (fase Uitvoering) voor een ander deel van de test al is gestart. Dit is een situatie die vaak voorkomt bij projecten waar de software gefaseerd wordt opgeleverd. Ook is het bijvoorbeeld aan te raden om tijdens de fase Specificatie al voorbereidingen te treffen voor de activiteiten in de fase Afronding. Dit fenomeen, dat fasen dus niet na elkaar uitgevoerd hoeven te worden, komt in het TMap faseringsmodel tot uitdrukking doordat de lijnen tussen de verschillende fasen schuin staan. Zo ontstaat de kenmerkende vorm van het model: een parallellogram.

Uitgediept

Hertesten binnen het TMap faseringsmodel

Binnen het faseringsmodel is ook plaats voor hertesten. Hertesten ontstaan doordat er bevindingen zijn geconstateerd tijdens het uitvoeren van de testgevallen. In die situatie dat er een hertest moet worden voorbereid en uitgevoerd, kan het nodig zijn enkele fasen van het TMap faseringsmodel opnieuw te doorlopen. Afhankelijk van de situatie kan dit beperkt blijven tot de fase Uitvoering, bijvoorbeeld in het geval er uitsluitend fouten in de software zijn opgelost. In het geval er fouten in de testbasis zijn opgelost, kan het noodzakelijk zijn om de hertest compleet te (her)plannen (zeker in het geval het een omvangrijke herstelactie van de testbasis betreft). Hierna worden dan de fasen Voorbereiding, Specificatie en Uitvoering opnieuw doorlopen.

Wanneer het faseringsmodel wordt gerelateerd met de systeemontwikkelfasering dan vallen een aantal relaties op. In figuur 6.2 “Relatie TMap testfasering met systeemontwikkelfasering” is een voorbeeld van deze relaties weergegeven.



Figuur 6.2: Relatie TMap testfasering met systeemontwikkelfasering.

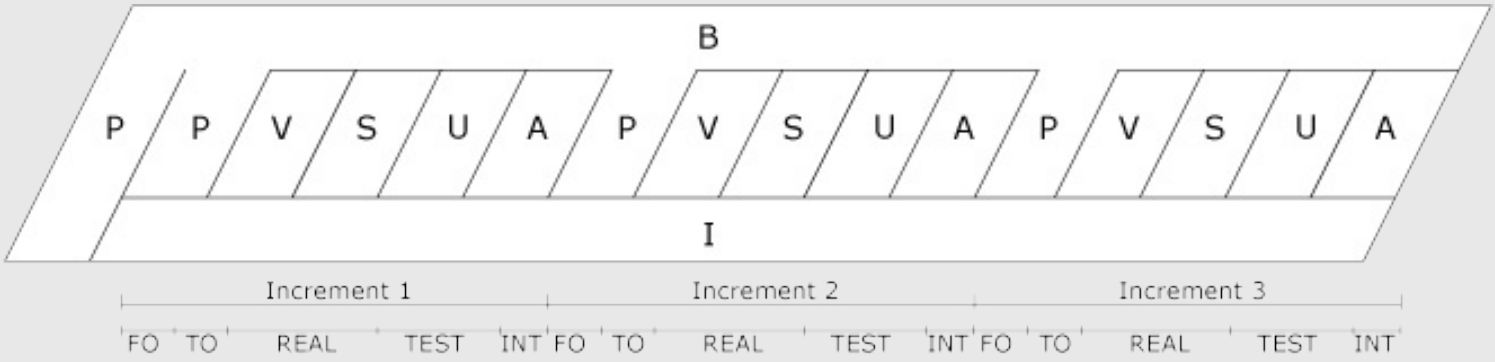
In bovenstaande figuur is te zien dat de voorbereidingsfase van de TMap testfasering kan starten wanneer de testbasis

wordt opgeleverd. De testbasis wordt opgesteld in de systeemontwikkelfasen FO (functioneel ontwerp) en/ of TO (technisch ontwerp). Na deze systeemontwikkelfasen start de realisatie van het testobject (de systeemontwikkelfase REAL). Zodra het testobject wordt opgeleverd start de test (TEST). De volgende systeemontwikkelfase is de implementatie (IMPL). Uit dit voorbeeld wordt duidelijk dat alleen de uitvoeringsfase van TMap zich op het kritieke pad van het project bevindt (het kritieke pad wordt weergegeven door de bolletjeslijn). Alle andere testfasen worden parallel aan de andere systeemontwikkelfasen uitgevoerd en bevinden zich, mits op tijd gereed, daarmee niet op het kritieke pad.

Uitgediept

TMap faseringsmodel in relatie tot verschillende modellen voor ontwikkeling

Het TMap faseringsmodel is toepasbaar binnen verschillende modellen voor systeemontwikkeling. Hierbij maakt het niet uit of de systeemontwikkeling gebeurt op basis van principes volgens bijvoorbeeld waterval, iteratief of incrementen. De reden hiervoor is dat elk model van systeemontwikkeling de systeemontwikkelingfasering kent zoals die in figuur 6.2 “Relatie TMap testfasering met systeemontwikkelfasering” staat beschreven. Bij iteratief en incrementeel ontwikkelen (bijvoorbeeld de methoden RUP en DSDM) moeten de eerste ontwikkelfasen uit het model (FO, TO en REAL) gezien worden als tussenproducten. Deze worden dan getest (TEST) en geintegreerd (INT). In figuur 6.3 “Relatie TMap testfasering met incrementen” wordt dat schematisch weergegeven.



Figuur 6.3: Relatie TMap testfasering met incrementen.

Op projectniveau, overkoepelend aan alle incrementen, worden de fasen Planning, Beheer en Inrichting en beheer infrastructuur uitgevoerd. Per increment worden de fasen Planning, Voorbereiding, Specificatie, Uitvoering en Afronding ingevuld. De fase Planning in de incrementen staat in nauwe relatie tot de overkoepelende fase Beheer, vandaar de open koppeling tussen deze twee. Vanwege het herhalende karakter van iteratief en incrementeel ontwikkelen moet wel veel nadruk worden gelegd op het herhaalbaar zijn van de tests. Dit kan door bijvoorbeeld testautomatisering en goed beheer van de testware.

6.2 Fase Planning

Doel

Het formuleren van een samenhangende en gedragen aanpak waarmee de testopdracht goed uitgevoerd kan worden. Een belangrijk onderdeel van de planning is het opstellen van het testplan, om de opdrachtgever en andere betrokkenen te informeren over de aanpak, planning, begroting, activiteiten en de op te leveren (eind)producten met betrekking tot het testproces. Indien er een bovenliggend mastertestplan is, dient dit plan hiervan afgeleid te worden.

Context

De stappen van de planningsfase dienen altijd doorlopen te worden. De resultaten worden meestal vastgelegd in een afzonderlijk testplan, wanneer de testsoort als op zichzelf staande activiteit wordt georganiseerd. In sommige gevallen, met name bij iteratieve of agile ontwikkeling, is de testsoort geïntegreerd in het totale proces en maakt het testplan onderdeel uit van het projectplan.

De benodigde inspanning voor het maken van het plan is afhankelijk van wat er al aanwezig is. De aanwezigheid van een mastertestplan, van Generieke Test Afspraken, of een lijnorganisatie Testen met voorschriften, sjablonen en standaards

kunnen het opstellen van het testplan aanzienlijk vergemakkelijken omdat hier simpelweg naar verwezen kan worden.

Bij het maken van het testplan moet de testmanager rekening houden met het (on)mogelijke. Een belangrijke factor hierbij is de bestaande “testvolwassenheid”, ofwel de kwaliteit van het testproces. Is men bekend met een testfasering, heeft men testtools, gebruiken de testers testontwerptechnieken, hoe is men gewend te administreren en te rapporteren? Bij lage testvolwassenheid kan de testmanager niet te hoge eisen stellen aan het testproces en de testers daarbinnen.

Dit geldt in mindere mate ook voor de volwassenheid van het ontwikkel- of onderhoudsproces rondom testen. Als dit chaotisch en onbeheersbaar is, is het vermoedelijk een slechte investering om het “perfecte” testproces neer te zetten en kan volstaan worden met een “redelijk” testproces.

Randvoorwaarden

Om zinvol te kunnen starten met het opstellen van het testplan dienen de volgende zaken bekend te zijn:

- De opdrachtgever voor de testsoort
- Doel en belang van het systeem of pakket voor de organisatie
- Globale eisen / requirements
- De organisatie van het ontwikkel-, onderhouds- of implementatieproces
- De (oplever)planning voor het te ontwikkelen of te onderhouden systeem of te implementeren pakket
- De werkwijze voor het te ontwikkelen of te onderhouden systeem of te implementeren pakket
- Indien sprake is van een mastertestplan dient dit gefixeerd en goedgekeurd te zijn
- Inzicht in ontwikkel- en productieomgeving voor het kunnen definiëren van de testomgeving.

Wanneer deze informatie nog niet beschikbaar is, bijvoorbeeld omdat de ontwikkelaanpak nog onbekend is, heeft dit negatieve gevolgen voor ofwel doorlooptijd of benodigde inspanning voor het opstellen van het plan, ofwel voor de kwaliteit en gewenste mate van detail van het plan. Ook moet er de bereidheid en mogelijkheid zijn tot het maken van afspraken op het gebied van de test.

Werkwijze

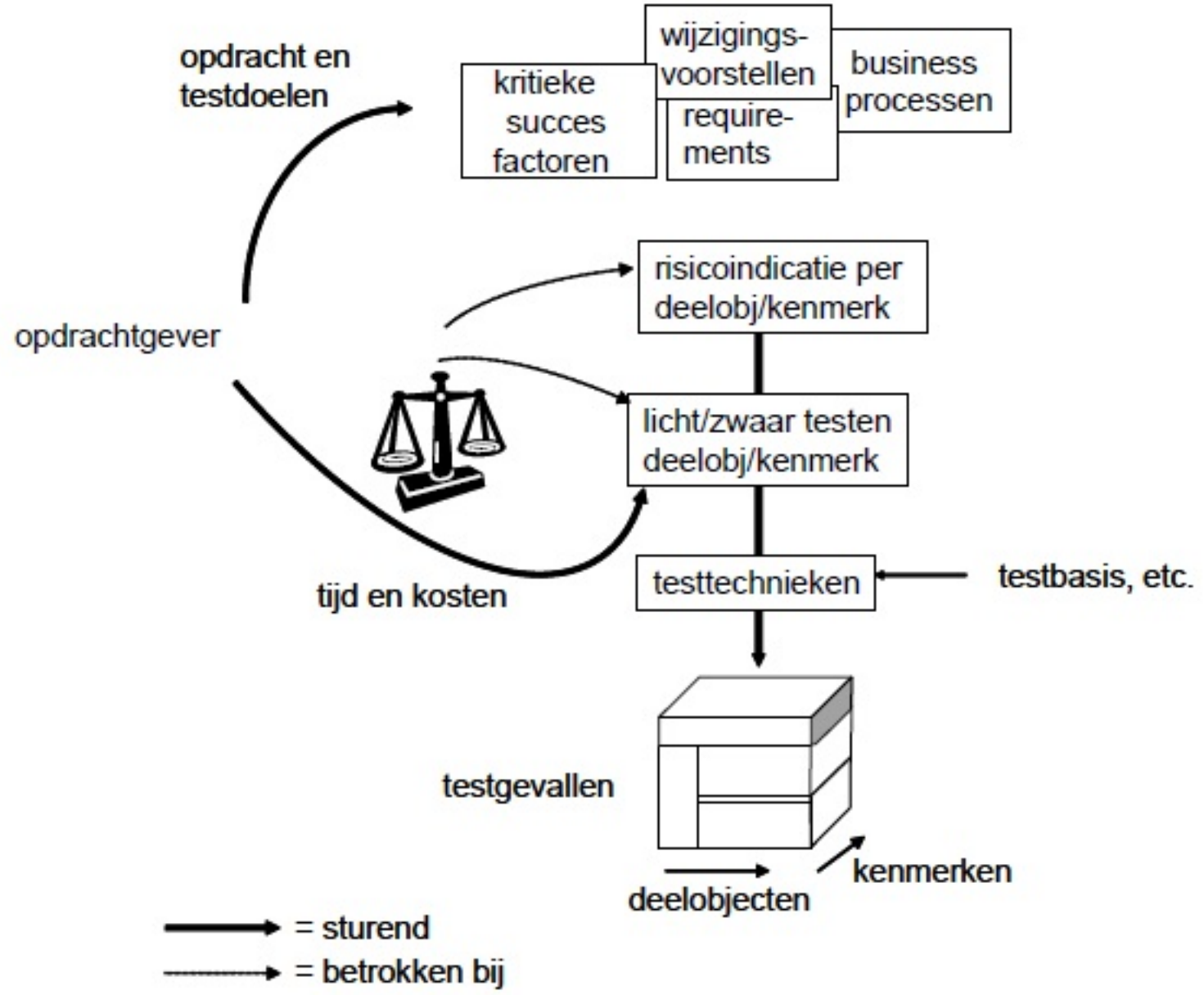
De testmanager is als regel de opsteller van het testplan. Met grote voorkeur is een mastertestplan aanwezig. Op basis hiervan en in samenspraak met de opdrachtgever formuleert hij de opdracht, rekening houdend met de vier BDTM-aspecten Resultaat, Risico’s, Tijd en Kosten (zie paragraaf 3.1 “Business driven toegelicht”). Vervolgens gaat de testmanager zich inwerken in het aankomende traject door diverse gesprekken met belanghebbenden te voeren en andere informatiebronnen zoals documentatie te raadplegen. Parallel hieraan vult hij in nauwe samenwerking met de opdrachtgever de opdracht verder in en bepaalt het beschouwingsgebied voor de testsoort.

Indien vanuit het mastertestplan nog geen productrisicoanalyse is uitgevoerd of deze te globaal is, wordt samen met de opdrachtgever en andere betrokkenen een gedetailleerde analyse uitgevoerd om op basis van de opdracht vast te stellen wat de gewenste resultaten van het testen voor de opdrachtgever moeten zijn (de *testdoelen*) en welke delen (*deelobjecten*) en *kenmerken* van het te testen systeem of pakket meer of minder risicovol zijn. Deze analyse vormt de basis voor de teststrategie.

Nu ontstaat een iteratief traject. In de strategie wordt op basis van de productrisicoanalyse bepaald welke kenmerken/deelobjecten getest moeten worden, met welke testvorm en met welke diepgang (hoe meer risico, hoe zwaarder). Vervolgens wordt de test op hoofdlijnen begroot en in een planning uitgezet (de grootste risico’s zo vroeg mogelijk beginnen af te dekken). Dit wordt afgestemd met opdrachtgever en andere betrokkenen en afhankelijk van hun oordeel mogelijk herzien. In dat geval worden de stappen dan opnieuw doorlopen. Conform BDTM heeft de opdrachtgever dus nadrukkelijk grip op het testproces en kan sturen in de balans Tijd en Kosten versus Resultaat en Risico.

Hierna vult de testmanager de strategie verder in door testeenheden te bepalen en de beslissingen over zwaar en minder zwaar testen te vertalen naar concrete uitspraken over welke dekking nagestreefd wordt. Vervolgens wijst hij *testtechnieken* toe aan de kenmerk/deelobject-combinaties, rekening houdend met de beschikbare testbasis, resources en infrastructurele voorzieningen. Met deze technieken worden in een later stadium de testgevallen (en bijvoorbeeld ook checklists) ontworpen en uitgevoerd.

Figuur 6.4 maakt dit duidelijk.



Figuur 6.4: Van opdracht en testdoelen naar testgevallen.

Verdere stappen in de planvorming zijn dat de testmanager de testbasis vaststelt, de testproducten definieert en de testorganisatie verder inricht. Er wordt op hoofdlijnen bepaald wat er aan infrastructuur moet komen. Testbeheer wordt ingericht met procedures en standaards, zoveel mogelijk ondersteund door tools. Hierbij geldt dat gebruik wordt gemaakt van wat al beschikbaar is, vanuit een mastertestplan, Generieke Test Afspraken, het testbeleid of de lijnorganisatie Testen.

De belangrijkste risico's die het testproces bedreigen worden benoemd en er worden mogelijke maatregelen voorgesteld om deze risico's te beheersen. Als laatste stap laat de testmanager het testplan goedkeuren door de opdrachtgever.

Hoewel de activiteiten in dit deelproces opvolgend beschreven zijn, zullen in de praktijk bepaalde activiteiten meerdere malen en/of in andere volgorde doorlopen worden. Wanneer bijvoorbeeld bepaalde voor een test benodigde infrastructuur niet geleverd kan worden, moet misschien de strategie aangepast worden.

Rollen/verantwoordelijkheden

De primair verantwoordelijke rol voor het opstellen van het testplan is de testmanager, soms ook testcoördinator genaamd.

Uitgediept

Testmanager of testcoördinator?

Hoewel in deze paragraaf consequent de term testmanager wordt gebruikt als verantwoordelijke persoon voor het testproces, staat in de praktijk ook vaak een testcoordinator aan het hoofd van de systeem- of acceptatietest. De verschillen zijn meer gevoelsmatig en situationeel dan objectief, maar in grote lijnen kan er het volgende over gezegd worden:

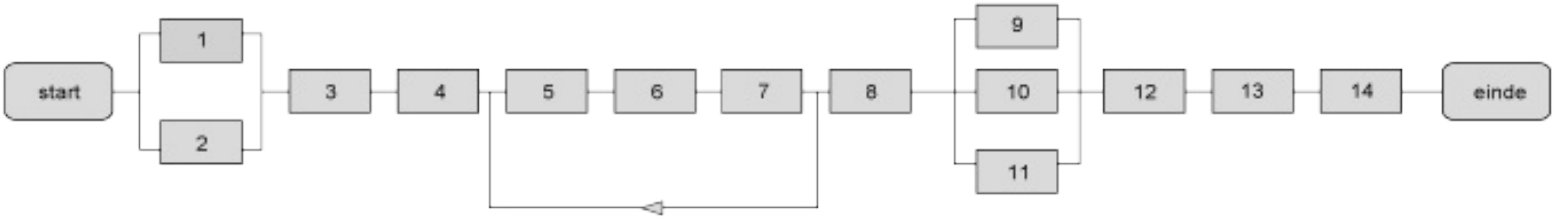
- Hoe meer bevoegdheden, hoe meer de voorkeur uitgaat naar de term testmanager;
- Hoe groter de scope van de test, idem;
- Hoe groter de omvang van de test, idem;
- Bij een overkoepelende testmanager: voorkeur testcoördinator;
- Bij een overall testcoördinator: voorkeur testmanager;

Activiteiten

Het opstellen van het testplan omvat de volgende activiteiten:

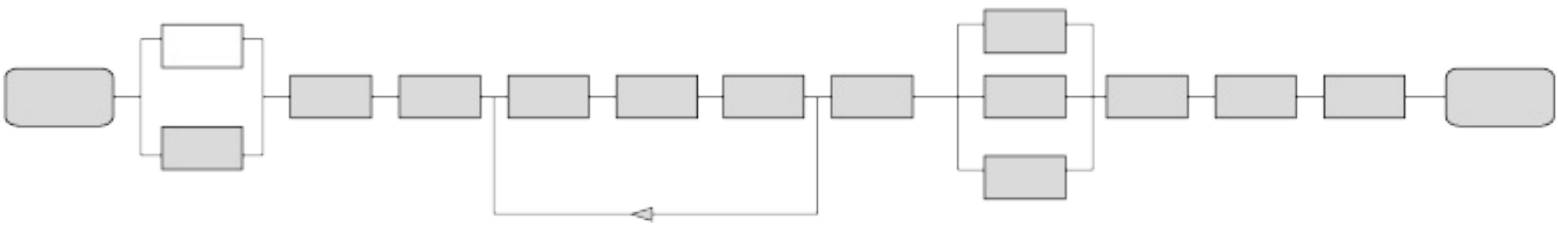
1. Vaststellen opdracht;
2. Oriënteren opdracht;
3. Vaststellen testbasis;
4. Analyseren productrisico's;
5. Bepalen teststrategie;
6. Bepalen begroting;
7. Bepalen planning;
8. Toewijzen testeenheden en testtechnieken;
9. Definiëren testproducten;
10. Definiëren organisatie;
11. Definiëren infrastructuur;
12. Inrichten beheer;
13. Bepalen testprocesrisico's en maatregelen;
14. Terugkoppelen en fixeren plan.

Onderstaand schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten. Voor alle activiteiten geldt dat deze meerdere malen doorlopen kunnen worden omdat de resultaten van een activiteit herziening van een voorgaande activiteit kunnen betekenen. Zoals al in de werkwijze is aangegeven, hebben de stappen 5, 6 en 7 een expliciet iteratief karakter:



Figuur 6.5: Opstellen testplan.

6.2.1 Vaststellen opdracht



Doel

Een systeem- of acceptatietest begint met een formulering van de opdracht zodat het doel, de taken en

verantwoordelijkheden van de testsoort voor alle betrokkenen duidelijk worden.

Werkwijze

Met het vastleggen van de opdrachtformulering in het testplan wordt duidelijk voor alle betrokkenen (waaronder de opdrachtgever) wat het testproces moet gaan opleveren. Verwachtingen worden over en weer op elkaar afgestemd.

De opdrachtformulering van de testsoort moet aansluiten op de opdrachtformulering zoals geformuleerd in het mastertestplan.

Een opdrachtformulering voor een testplan bestaat uit de volgende onderdelen:

- Opdrachtgever
- Opdrachtnemer
- Opdracht
- Beschouwingsgebied
- Randvoorwaarden en uitgangspunten

Deze onderdelen worden hieronder nader toegelicht:

Opdrachtgever

De partij die de opdracht geeft tot het opstellen van het testplan en het uitvoeren van de tests. Het is belangrijk voor de testsoort om te onderkennen wie de opdracht geeft voor het uitvoeren van de test.

Uitgediept

In de praktijk zien we voor de verschillende testsoorten als regel de volgende mogelijkheden:

Systeemtest	• Projectmanager vanuit leverancier • Projectmanager/-leider van realisatie
Functionele acceptatietest	• Projectmanager vanuit klant/acceptanten • Hoofd functioneel beheer
Systeemintegratietest	• Projectmanager vanuit klant/acceptanten • Hoofd functioneel beheer
Gebruikersacceptatietest	• Projectmanager vanuit klant/acceptanten • Hoofd gebruikersorganisatie
Productieacceptatietest	• Projectmanager vanuit klant/acceptanten • Hoofd systeembeheer

Opdrachtnemer

Normaal gesproken is een testmanager of testcoördinator verantwoordelijk voor het opstellen van het testplan en de uitvoering van de testopdracht.

Opdracht

Deze dient in overleg met de opdrachtgever te worden opgesteld. Hierin dient de doelstelling van de test en een afbakening van de opdracht te worden aangegeven.

Uitgediept

Het lijkt alsof dit onderdeel de vanzelfsprekende kern is van de activiteit “Vaststellen opdracht”. Hoewel niet onbelangrijk, is de opdrachtformulering in de praktijk vaak enigszins abstract en generiek geformuleerd in termen van “kwaliteitsoordeel geven” of “inzicht geven in risico’s”. Het zijn met name de onderdelen beschouwingsgebied, randvoorwaarden en uitgangspunten (en later de strategie) waar de totale opdracht scherp wordt neergezet.

Iteraties

Iteratieve of agile systeemontwikkeling leveren een flink aantal te testen (tussen) releases of prototypes op. Uit de opdrachtformulering moet duidelijk blijken dat een dergelijke tussenrelease of prototype niet op alle aspecten van een komend productiesysteem beoordeeld mag worden, maar alleen op de aspecten die relevant zijn voor de tussenrelease of prototype zelf.

Het is aan te bevelen om als testmanager al voeling te krijgen waar het project primair op gestuurd gaat worden in termen van BDTM. Gaat de opdrachtgever sterk op Tijd of op Kosten sturen, of is Resultaat/Risico leidend? Dit is geen gemakkelijke opgave, want de eerste reactie ("ons maximale budget is € …", "de deadline van … is in beton gegoten") lijkt vaak kristalhelder, maar blijkt dat bij doorvragen niet altijd te zijn ("… en als het systeem dan maar 3⁄4 van de functionaliteit heeft?"). Niettemin draagt dit inzicht bij aan de noodzakelijke beeldvorming voor de testmanager en vergemakkelijkt het latere communicatie over te maken keuzes. De gevoeligheid van deze informatie maakt dat dit niet in het plan wordt vastgelegd!

Daarnaast krijgt de test regelmatig een secundaire opdracht. Hiervoor dient de opdrachtgever aanvullend budget en/of tijd ter beschikking te stellen.
Voorbeelden zijn:

- maken van een *standaard onderhoudstestplan* met daarin alle herbruikbare testaspecten;
- opleiden en coachen van de medewerkers in het testvak;
- verbeteren en structureren van de gehanteerde testaanpak;
- implementeren van een testtool;
- opzetten, gebruiken en onderhouden van een schaalbare regressietestset (zie paragraaf 6.6 "Fase specificatie");
- opleveren van (geautomatiseerde) testware voor het testen van volgende releases.

Uitgediept

Normaal gesproken stelt de opdrachtgever resources (mensen en middelen) beschikbaar of betaalt daarvoor, bijvoorbeeld door interne of externe mensen in te huren. Verrekening gaat meestal op uur-basis. In bepaalde gevallen, met name bij outsourcing wanneer de test aan een externe leverancier is uitbesteed, kunnen creatievere afspraken worden gemaakt. Onderstaand een aantal mogelijke constructies die in de praktijk voorkomen:

- Fixed-price
  De leverancier voert de test uit tegen een vooraf bepaalde, vaste prijs. Hierbij is meestal een vast aantal hertests inbegrepen. Bij verstoring van het testproces omdat de opdrachtgever niet kan voldoen aan de gestelde uitgangspunten óf wanneer meer (her)tests nodig zijn dan afgesproken, wordt meerwerk in rekening gebracht. In de andere gevallen is het risico voor de leverancier.
- Fixed-price per testgeval
  Een variant op bovenstaande is dat er een vast bedrag per te specificeren en uit te voeren testgeval wordt afgesproken.
- Fixed-date
  Idem als fixed-price, maar dan met een vaste einddatum
- Fixed-date, fixed-price
  Idem, met zowel vaste prijs als einddatum
- Bonus-malus
  Aanvullend op bovenstaande kunnen afspraken gemaakt worden om het risico beter te verdelen over beide partijen. Hierbij betaalt de opdrachtgever de leverancier per uur, maar gelden wel fixed-date of -price. Wanneer de leverancier minder uren of doorlooptijd nodig heeft, krijgt deze een bonus in de vorm van meer geld. Voorbeeld van malus: als binnen X tijd na in-productie-name kritische fouten optreden of bij overschrijding van tijd of uren geeft de leverancier korting op de tarieven.
- Resultaatdeling
  Een onconventionele vorm is wanneer de leverancier betaald wordt met een percentage van de opbrengsten van het nieuwe systeem. Hierbij is het systeem voor zowel leverancier als opdrachtgever een investering en hebben beiden alle belang bij een goed eindresultaat. Dat hier grote risico's (maar ook kansen) aan kleven mag duidelijk zijn.

Beschouwingsgebied

Hier dient aangegeven te worden wat de grenzen zijn van het beschouwingsgebied van de test. Dit moet bij voorkeur specifieker dan wat al in het mastertestplan is opgenomen. De volgende zaken moeten worden meegenomen (indien van toepassing):

- syste(e)m(en);
- conversies;
- administratieve organisatie (AO)-procedures;
- kwaliteitsattributen (toegewezen in het mastertestplan);
- interfaces met aangrenzende systemen (wordt de interface getest *tot* het andere systeem of *tot en met* of zelfs tot en met de hele keten?).

Bij wijzigingen moet bepaald worden welke delen van bovenstaande beschouwd worden.
Daarnaast wordt aangegeven welke zaken buiten de scope van het testen vallen. Naast bovengenoemde zaken moet men denken aan:

- systeemwijzigingen die niet in het project worden meegenomen;
- testactiviteiten die door andere testsoorten of partijen worden uitgevoerd;
- reorganisaties;
- mogelijk toekomstige projecten die op het huidige project van invloed zijn (met name als er over andere projecten nog geen duidelijkheid is).

Randvoorwaarden

Onder randvoorwaarden worden voorwaarden beschreven die derden zoals opdrachtgever, het project, beheerders of gebruikers, aan het testproces opleggen en waarbinnen het testproces moet opereren.

Voorbeeld

- Mastertestplan
  Het mastertestplan is sturend voor het inrichten en uitvoeren van de testsoort;
- Mijlpalen
  Op het moment dat de testopdracht wordt verstrekt, is vaak al een aantal mijlpalen vastgesteld, zoals de opleverdatum van de testbasis, het testobject en de infrastructuur en de datum van in-productie-name;
- Beschikbare resources
  Door de opdrachtgever zijn er vaak grenzen gesteld aan de beschikbare mensen, middelen en budget;
- Te hanteren normen en standaards
  Vanuit de (test)organisatie of het mastertestplan kunnen bepaalde eisen zijn gesteld ten aanzien van werkwijze, procedures, technieken, sjablonen, enzovoort.

Uitgangspunten

Dit zijn externe omstandigheden of gebeurtenissen die moeten gebeuren om het testproces succesvol te laten zijn, maar die buiten de controle van het testproces vallen. Anders gezegd: de eisen die het testproces stelt aan de anderen.

Voorbeeld

- **Kwaliteit van voorgaande tests**
  De voorgaande tests, bijvoorbeeld ontwikkel- of systeemtests zijn op de afgesproken wijze uitgevoerd;
- **Kwaliteit van testobject**
  Het testobject heeft de afgesproken ingangskwaliteit. Dit moet geconcretiseerd worden met behulp van zogenaamde entry-criteria. Deze hebben overlap met (maar zijn niet noodzakelijk hetzelfde als) de exitcriteria van de voorgaande test;
- **Te leveren ondersteuning**
  Vanuit het testproces is er behoefte aan verschillende vormen van ondersteuning, bijvoorbeeld ten aanzien van de testbasis, testobject, materiedeskundigheid en/of de infrastructuur. Deze ondersteuning kan in een bepaalde

omvang gewenst zijn en/of gedurende een bepaalde periode. Denk bijvoorbeeld aan de beschikbaarheid van ontwikkelaars tijdens de testuitvoering om blokkerende fouten op te lossen. Normaal gesproken heeft elke testsoort een eigen expertise, bijvoorbeeld de gebruikersacceptatietest zal weinig behoefte hebben aan materiedeskundige ondersteuning, en geldt de behoefte aan ondersteuning juist de andere soorten expertise, zoals technische of testmethodische ondersteuning.

- **Wijzigingen in testbasis en testobject**
  Het testteam moet betrokken worden bij het doorvoeren van wijzigingen. In de meeste gevallen betekent dit eenvoudigweg het aansluiten op de bestaande procedures binnen het systeemontwikkelproces. Zo moet de testmanager deel uitmaken van het Change Control Board om vanuit testoptiek de gevolgen van een wijziging in te schatten.
- **Oplevering van het testobject**
  Het bouwteam levert het testobject op in een aantal verschillende maar efficiënt testbare delen en draag zorg voor installatie in de testomgeving.
- **Reactietijd op bevindingen**
  Hoe snel moet het project reageren op bevindingen. Onderstaand een voorbeeld van dergelijke afspraken:

Ernst	Prioriteit	Reactietijd	Doorlooptijd
Testblokkerend	Hoog	1 uur	4 uur
Ernstige fout	Hoog	1 uur	1 werkdag
Fout	Hoog	1 werkdag	2 werkdagen
Fout	Laag	1 werkdag	Per fout te bepalen
Wens	Laag	2 werkdagen	Per wenste bepalen

De testmanager kan er niet mee volstaan om deze punten in het plan op te nemen en er dan bij acceptatie van het plan van uit te gaan dat alle punten geregeld zijn. Integendeel, hij moet eerst de punten afstemmen met de partijen die hier zeggenschap over hebben, zodat de punten een vastlegging van afspraken zijn en geen verrassing. Het is aan te bevelen in het plan per uitgangspunt op te nemen voor welke partij(en) deze bedoeld is.

Voor een checklist van mogelijke randvoorwaarden en uitgangspunten wordt verwezen naar www.tmap.net.

Producten

De opdrachtformulering, vastgelegd in het testplan.

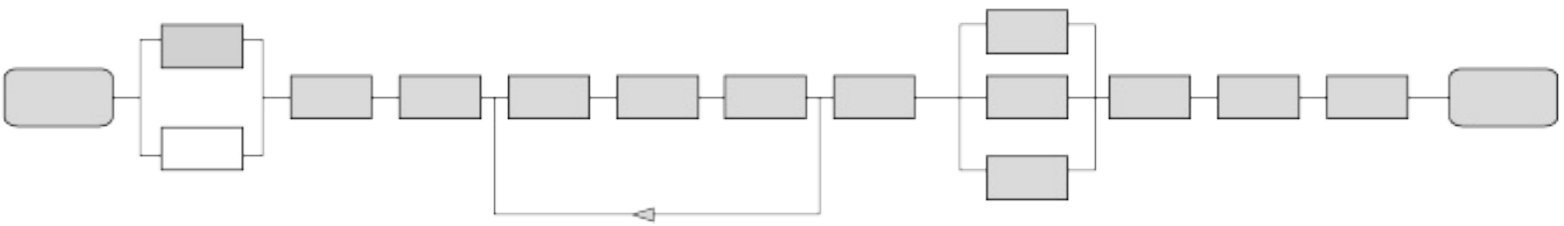
Technieken

Checklist randvoorwaarden en uitgangspunten (www.tmap.net).

Tools

Niet van toepassing.

6.2.2 Oriënteren opdracht



Doel

Het verkrijgen van inzicht in de (project)organisatie, de doelstelling en opzet van het systeemontwikkelproces, het te testen systeem of pakket en de eisen waaraan dit moet voldoen, zodat er beter richting gegeven kan worden aan de

overige stappen van de planvorming.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen acceptanten met acceptatiecriteria en overige informatiegevers;
2. Het bestuderen van de beschikbare documentatie;
3. Het afnemen van interviews.

In de praktijk wordt deze activiteit parallel aan de opdrachtformulering uitgevoerd en ook enigszins onderschat. De onderschatting bestaat er met name uit dat de testmanager met te weinig belanghebbenden spreekt, terwijl het juist in het begin essentieel is om de verwachtingen goed te peilen en als testmanager overal de “voelsprieten uit te steken”. Dit is nodig voor het goed kunnen uitvoeren van volgende activiteiten én voor het in de toekomst goed kunnen beheersen van het testproces.

1. Bepalen acceptanten met acceptatiecriteria en overige informatiegevers

Meestal is de opdrachtgever niet de enige partij die het systeem moet accepteren, maar zijn er meerdere acceptanten. Het is van belang duidelijk te krijgen wie deze accepterende partijen zijn. Dit gebeurt in samenspraak met de opdrachtgever. In de praktijk heeft de testmanager hier de kans om belanghebbenden op hoog niveau in de organisatie (stuurgroepleden) een keer te spreken en hun mening en verwachting te peilen. Later komt dit er vaak niet meer van, tenzij de testmanager in de (helaas) vrij zeldzame positie verkeert dat hij regelmatig deelneemt aan het stuurgroepoverleg.

Vastgesteld dient te worden welke acceptanten tijdens het project direct of indirect middels testrapportages van informatie voorzien dienen te worden. Daarbij moet duidelijk zijn wat elke acceptant aan concrete eisen of acceptatiecriteria stelt. Dit zijn kwalitatieve eisen waar het product en de omgeving minimaal aan moeten voldoen om het acceptabel te laten zijn voor de acceptant. Voor de duidelijkheid: het verzamelen van acceptatiecriteria is geen verantwoordelijkheid van de testers, maar wel input voor het in te richten testproces. Acceptatiecriteria kunnen behoorlijk uiteenlopen. Enkele voorbeelden zijn:

- kwalitatieve criteria aan het product en voortbrengingsproces, bijvoorbeeld het aantal bevindingen dat open mag staan;
- criteria aan de omgeving, bijvoorbeeld de infrastructuur moet geïnstalleerd zijn of de gebruikers moeten training gevolgd hebben;
- criteria in de vorm van (detaillering van) requirements aan het product, bijvoorbeeld “een order moet binnen X seconden worden verwerkt”

Niet alle acceptatiecriteria zijn relevant voor testen. Het eerste voorbeeld heeft grote overlap met de in paragraaf 6.2.7 “Bepalen planning” besproken exitcriteria voor het testproces. Het tweede voorbeeld is meestal minder van belang voor testen, het derde voorbeeld is een vorm van testbasis.

Uitgediept

Valkuil acceptatiecriteria

Dit laatst genoemde gebruik van acceptatiecriteria houdt een gevaar in. In de praktijk komt het volgende nogal eens voor: na het vastleggen en bevriezen van de requirements komen gebruikers er achter dat ze nog meer wensen hebben. Deze wensen formuleren ze dan als acceptatiecriteria. Acceptatiecriteria vormen op deze manier het “achterdeurtje” om alsnog requirements op te nemen. Dit is geen goede werkwijze. De enige juiste manier is het indienen van een wijzigingsvoorstel bij een Change Control Board.

Naast acceptanten zijn er diverse andere partijen/personen die voor het testproces relevante informatie kunnen geven. Denk hierbij aan:

Uitgediept

- de testmanager op overkoepelend niveau, om inzicht te krijgen in de testopdracht en wat er van de test(manager) verwacht wordt;

- de (vertegenwoordigers van de) opdrachtgever, om inzicht te krijgen in zowel de bedrijfsdoelstellingen en de "cultuur" als de doelstelling en het strategisch belang van het systeem ;
- de projectmanager of kwaliteitszorgmedewerker, om inzicht te krijgen in de stappen en onderdelen van het ontwikkelproces en de onderlinge samenhang, met speciale aandacht voor de (verwachte) plaats van testen hierin;
- de materiedeskundigen uit de gebruikersorganisatie, om inzicht te krijgen in de (benodigde) functionaliteit van het systeem;
- de ontwerpers, om inzicht te krijgen in de te ontwikkelen functionaliteit van het systeem;
- systeembeheerders, om inzicht te krijgen in de (toekomstige) productie-omgeving van het informatiesysteem;
- testers, om inzicht te krijgen in de testaanpak en testvolwassenheid van de organisatie;
- de leveranciers van de testbasis, het testobject en de infrastructuur, om in een vroegtijdig stadium reeds afstemming tussen de verschillende betrokken partijen te garanderen.

2. Het bestuderen van de beschikbare documentatie

De door de opdrachtgever beschikbaar gestelde documentatie wordt bestudeerd. Denk hierbij aan:

Uitgediept

- testdocumentatie zoals het mastertestplan of een document Generieke Test Afspraken;
- systeemdocumentatie, zoals stakeholderanalyse, business of user requirements of een informatieanalyse, system requirements, functioneel en technisch ontwerp;
- projectdocumentatie, zoals het plan van aanpak voor het systeemontwikkelproces, organisatieschema's en verantwoordelijkheden, het kwaliteitsplan, reviewverslagen en een functiepuntanalyse;
- een beschrijving van de systeemontwikkelmethode inclusief de normen en standaards;
- een beschrijving van de gehanteerde testmethode inclusief de normen en standaards;
- evaluaties en leerpunten van vorige testtrajecten die relevant kunnen zijn voor de komende test;
- contracten met leveranciers.

Als het systeemontwikkelproces betrekking heeft op onderhoud, dan wordt eveneens een onderzoek gedaan naar de aanwezigheid en bruikbaarheid van bestaande testware.

3. Het afnemen van interviews

De diverse betrokkenen bij het systeem ontwikkel proces worden geïnterviewd.

De testmanager interviewt de betrokkenen, naast algemene achtergrondvragen over het te testen systeem en te volgen traject, over:

- wat verwacht de geïnterviewde als resultaat van het testen, wat wil deze als eindresultaat zien? Denk hierbij aan door IT ondersteunde bedrijfsprocessen, gerealiseerde user requirements of use cases, wijzigingsvoorstellen, kritische succesfactoren, benoemde (af te dekken) risico's maar bijvoorbeeld ook dat het nieuwe systeem minimaal exact dezelfde functionaliteit heeft als het oude systeem (dus geen regressie). Dit worden de testdoelen genoemd. Sluiten deze aan op de testdoelen uit het mastertestplan?

Definitie

Een testdoel is een voor de opdrachtgever relevant doel voor het testen, vaak geformuleerd in termen van door IT ondersteunde bedrijfsprocessen, gerealiseerde user requirements of use cases, kritische succesfactoren, wijzigingsvoorstellen of benoemde (af te dekken) risico's.

- heeft de geïnterviewde een beeld wat de kenmerken (meestal de kwaliteitsattributen) en deelobjecten zijn die spelen bij bovenstaande? Sommige personen kunnen dit prima beantwoorden, voor anderen gaat dit mogelijk te diep. De testmanager moet de inschatting maken hoe gedetailleerd erover gesproken kan worden.
- wat is de eventuele testbasis die straks als bewijs gebruikt kan of moet worden dat er voldoende getest is?
- wat is de risico-inschatting van de testdoelen en/of kenmerken/deelobjecten? Een risico wordt hierbij gedefinieerd als

het product van Schade x (Foutkans x Gebruiksfrequentie). Vaak zal de geïnterviewde van een risico alleen bepaalde aspecten kunnen noemen, zoals de Schade, de Frequentie van gebruik óf de Foutkans. Dat geeft niet, want in de volgende stap, de productrisicoanalyse, kan het verder aangevuld worden.
- wil betrokkene gerapporteerd worden en zo ja, op welk niveau? Aandachtspunt hier is dat het aantal soorten rapportage om praktische redenen enigszins beperkt moet blijven. Zo nodig moet de testmanager dit overleggen met de opdrachtgever.

Tevens is het raadzaam waar mogelijk indirect betrokkenen te raadplegen. Denk hierbij aan de accountantsdienst, de implementatiemanager, de toekomstige onderhoudsorganisatie, enzovoort.

Tip

In plaats van afzonderlijke interviews kan ook een kick-off sessie met (een aantal van) de relevante partijen georganiseerd worden. Voordeel hiervan is dat de verschillende gezichtspunten helpen om gezamenlijk tot een helder beeld te komen. Met name bij onderhoud, om de (test)impact van wijzigingsvoorstellen te bepalen, gebeurt dit vaak.

De testmanager koppelt de bevindingen van deze activiteit ter verificatie terug met de opdrachtgever.

Producten

Deze activiteit levert de onderstaande onderdelen van het testplan:

- Belanghebbenden en acceptatiecriteria
  De voor het testen relevante belanghebbenden en hun acceptatiecriteria.
- Normen en standaards
  Hier worden de gebruikte standaards genoemd. Wat betreft testen kan daarbij gedacht worden aan voorschriften van de lijnorganisatie Testen, het mastertestplan of generieke test afspraken, aan TMap, TPI of testhandboeken. Ontwikkelstandaarden, documentstandaarden of kwaliteitsnormen die gevolgd (moeten) worden, kunnen hier ook worden opgenomen.
- Basis voor het testplan
  Hier worden de documenten genoemd die de basis vormen voor dit testplan. Denk hierbij aan een mastertestplan, projectplan, specifieke project- of testplanningen, een specifieke of een generieke testmethode, generieke test afspraken, een implementatieplan of andere documenten die van belang zijn.

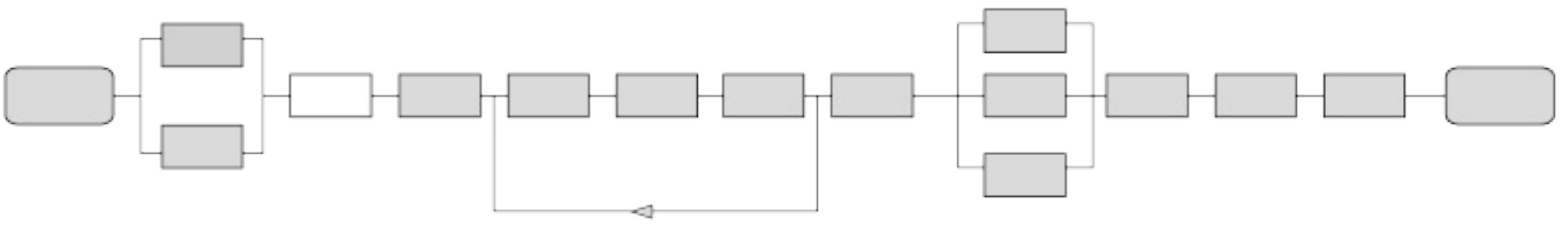
Technieken

Checklist “Oriënteren testopdracht” (www.tmap.net).

Tools

Niet van toepassing.

6.2.3 Vaststellen testbasis



Doel

Het eenduidig definiëren van de testbasis zodat vroegtijdig bekend is waar het testobject mee vergeleken moet worden.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen testbasis;
2. Identificeren testbasis.

1. Bepalen testbasis

De testbasis, ofwel de verzameling van alle geschreven en ongeschreven eisen waaraan het testobject moet voldoen, kan op veel verschillende manieren worden vormgegeven. Denk hierbij bijvoorbeeld aan requirements, acceptatiecriteria, functioneel ontwerpen, technisch ontwerpen, gebruikershandleidingen, interviews, verslagen van bijeenkomsten, wetgeving, maar ook aan het oude systeem, een vorige release van het systeem, een prototype of zelfs een materiedeskundige.

Met name het verzamelen van niet-gedocumenteerde testbasis is lastig, meer informatie hierover is te vinden in paragraaf 6.5 “Fase Voorbereiding”.

Het is bij het bepalen van de testbasis belangrijk om zeker te stellen dat ook de niet-functionele eisen bekend zijn, zoals bijvoorbeeld eisen ten aanzien van performance of beveiliging.

Uitgediept

Veelal maken de afzonderlijke testsoorten gebruik van verschillende bronnen waaruit zij de producteisen halen waartegen in die testsoort wordt getest. Zo is de acceptatietest veelal gericht op eisen die op het niveau van bijvoorbeeld bedrijfsprocessen zijn beschreven. In de unittest wordt juist gecontroleerd of aan de technische eisen voor een specifieke unit is voldaan. De testbasis zal daarom niet voor alle testsoorten gelijk zijn.

Opmerking bij de testbasis is dat deze eisen zo concreet en meetbaar (testbaar!) mogelijk zijn, waardoor er geen misverstanden over kunnen ontstaan. In de praktijk is dat vaak niet het geval. Wanneer mogelijk kan dat hier al gesignaleerd worden, anders wordt het in de latere fasen Voorbereiding en Specificatie alsnog gesignaleerd. Ook kan het zijn dat in het verdere traject blijkt dat een eis heel moeilijk testbaar is. In dergelijke gevallen wordt op dat moment met de opdrachtgever afgestemd of een versimpelde test acceptabel is.

2. Identificeren testbasis

Stel, voor zover mogelijk, de identificatie van de relevante testbasis vast. Denk hierbij aan opleverdatum, versie, status, enzovoort.

Producten

De te hanteren testbasis, vastgelegd in het testplan.

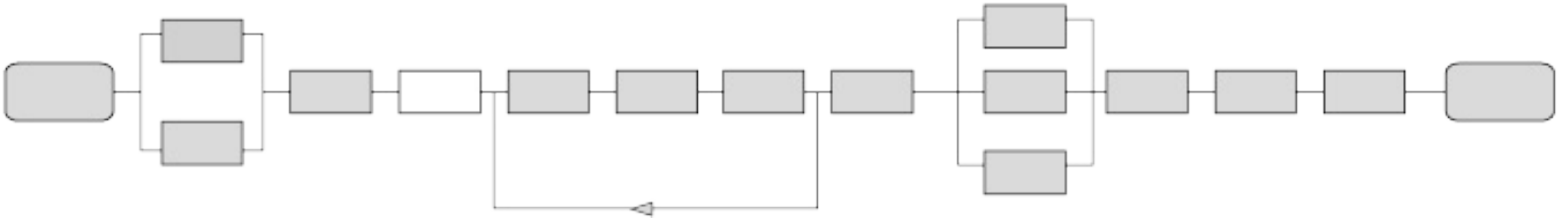
Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.2.4 Analyseren Productrisico’s



Doel

Het tot een gezamenlijk beeld komen van de verschillende deelnemers en de testmanager van wat de meer of minder risicovolle delen en kenmerken van het systeem zijn.

Werkwijze

Testen is een maatregel om inzicht te geven in de kwaliteit en daaraan gerelateerde productrisico's van een systeem of pakket bij in-productie-name voor een organisatie. Aangezien er nooit sprake zal zijn van een onbeperkte hoeveelheid middelen en tijd is het belangrijk om zo vroeg mogelijk te bepalen welke systeemdelen of -kenmerken extra of juist minder testinspanning vragen. Hiervoor moeten onderbouwd keuzes gemaakt worden. Hulpmiddel bij het bepalen van aandachtsgebieden voor de test is het uitvoeren van een productrisicoanalyse (PRA).

Een PRA in het kader van een testsoort is optioneel: wanneer de mastertestplan-PRA al met voldoende detail (dat wil zeggen op het niveau van kenmerken en deelobjecten) is uitgevoerd, kan deze stap overgeslagen worden. Is er geen mastertestplan-PRA, dan moet eerst bepaald worden wat de testdoelen zijn en wat hun relatie is naar de kenmerken/deelobjecten welke getest gaan worden in de betreffende testsoort. Dit gaat op eenzelfde manier als de PRA voor het mastertestplan.

Is er wel een mastertestplan-PRA maar is deze niet op het niveau van deelobjecten, dan moet de PRA verder uitgewerkt worden.

Het uitvoeren van een productrisicoanalyse valt uiteen in de volgende subactiviteiten:

1. Bepalen deelnemers
2. Bepalen van de PRA-aanpak
3. Voorbereiden sessie/interviews
4. Verzamelen en analyseren productrisico's
5. Volledigheidscontrole

In hoofdstuk 9 "Productrisicoanalyse" worden deze stappen gedetailleerd toegelicht.

Tips

- Aandachtspunt is dat de eventuele PRA-sessie mogelijk gecombineerd kan worden met (een deel van) de volgende activiteit, Bepalen teststrategie.
- Bij een PRA voor een testsoort is de uitdaging om de testdoelen, kenmerken en deelobjecten enkel betrekking te laten hebben op het beschouwingsgebied van de testsoort. Het heeft weinig zin om beveiliging als groot risico te onderkennen als dit kenmerk vanuit het mastertestplan al is toegewezen aan een andere testsoort.

Producten

De risicotabellen met per kenmerk testdoelen en mogelijke deelobjecten met risico-indicaties, apart beheerd en optioneel vastgelegd in het testplan.
Het PRA-totaaloverzicht met kenmerken/deelobjecten met risicoklasse, vastgelegd in het testplan.

Technieken

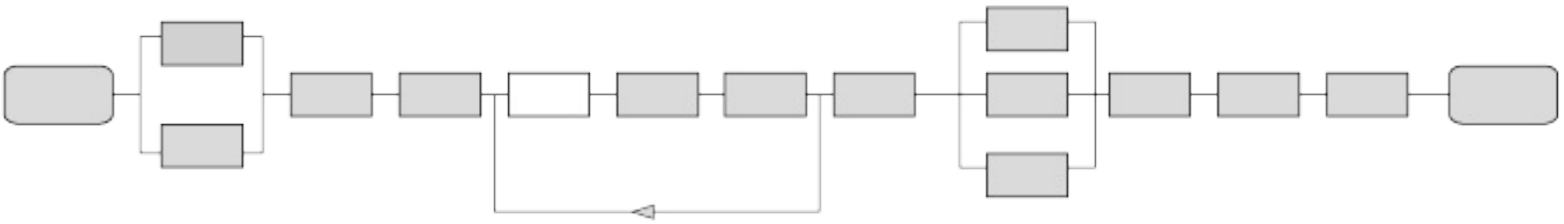
Productrisicoanalyse (hoofdstuk 9).

Uitleg kwaliteitsattributen (hoofdstuk 10).

Tools

Niet van toepassing.

6.2.5 Bepalen teststrategie



Doel

Het op basis van het inzicht in de meer of minder risicovolle deelobjecten/ kenmerken van het systeem maken van een keuze welke testvormen gehanteerd worden en hoe licht of zwaar getest moet worden voor elke (combinatie van) kenmerk/deelobject van het systeem.

Werkwijze

In de strategiebepaling voor een testsoort wordt de keuze gemaakt voor testvormen en de zwaarte van testen, ofwel hoe licht of zwaar worden de combinaties van kenmerken en deelobjecten getest. Dit is afhankelijk van de risicoinschatting uit de PRA, of liever de mate waarin opdrachtgever deze risico's wil afdekken en hoeveel tijd/geld deze hiervoor beschikbaar stelt.

Hiertoe doet de testmanager een voorstel bij elke combinatie van deelobject/ kenmerk voor de gewenste testvormen en voor de testzwaarte.

Testvormen

Hiermee maakt de testmanager duidelijk wat er van een bepaalde kenmerk/ deelobject-combinatie getest gaat worden. Op z'n simpelst is dit de test van een kwaliteitsattribuut, bijvoorbeeld functionaliteitstest of performancetest, maar vaak kan én moet al meer inzicht gegeven worden. Zo zijn andere testvormen horend bij het kwaliteitsattribuut functionaliteit bijvoorbeeld multiuser test, regressietest of ketentest. Op www.tmap.net is een overzicht "Toegepaste testvormen" opgenomen.

Uitgediept

Traditioneel wordt de meeste aandacht besteed aan het testen van functionaliteit van het systeem. De IT laat echter steeds meer een verschuiving in het testen zien naar andere kenmerken, zoals inpasbaarheid, beveiliging, portabiliteit, performance en usability. Met name het internet heeft ervoor gezorgd dat deze kenmerken steeds belangrijker én risicovoller zijn geworden. Deze kenmerken kunnen net als functionaliteit getest worden door het toepassen van testontwerptechnieken of checklists. Daarnaast zijn er echter voor deze kenmerken specifieke aandachtspunten en daaraan gerelateerde manieren om ze te testen. In paragraaf 10.3 "Testvormen" staat daarom een beschrijving van deze kenmerken en de hieraan gerelateerde testvormen.

Testzwaarte

Voor de testzwaarte wordt een keuze gemaakt uit onderstaande mogelijkheden:

- zware dynamische test

●● gemiddelde dynamische test
● beperkte dynamische test
S statisch testen
Controleren en onderzoeken van producten zonder dat software wordt gerund
I Impliciet testen
Mee testen bij een andere testvorm zonder het maken van expliciete testgevallen, alleen opvallende bevindingen worden vastgelegd
- Als in een cel niets staat, betekent dit dat de betreffende toets- of testsoort geen aandacht hoeft te besteden aan het kenmerk.

Uitgediept

Het resultaat van deze stap ziet er als volgt uit:

Voorbeeld ST

Kenmerk	RK MTP	RK MTP		Deelsys1	Deelsys2	Totaalsys
Functionaliteit	n.v.t.	n.v.t.		A/●●● functionele, regressie	B/●● functionele, regressie	C/● integratie, multi-user
Performance online	B	●		-	-	C/● steekproef in ST-omgeving
…						

RK MTP = Vanuit het mastertestplan toegekende risicoklasse aan het kenmerk
ST MTP = Vanuit het mastertestplan toegekende testzwaarte aan de test (in dit geval de systeemtest)
n.v.t. = in het mastertestplan zijn risicoklasse en zwaarte niet op kenmerkniveau toegekend, maar op het niveau van kenmerk/deelobject
A = Hoge risicoklasse
B = Gemiddelde risicoklasse
C = Lage risicoklasse

De risicoklassen kunnen worden overgenomen uit de PRA van het mastertestplan of (in meer detail) uit de PRA van de testsoort zelf.

De testmanager voorziet deze tabel vervolgens van de nodige toelichting. In bovenstaande voorbeeld gaat voor deelsysteem 1 en 2 een functionele test uitgevoerd worden op de nieuwe en gewijzigde functionaliteit, plus wordt een regressietest op ongewijzigde delen voorzien. Na het afzonderlijk testen van deelsystemen 1 en 2 wordt het totale systeem op integratieaspecten getest én vindt nog een multi-user test plaats. Performance wordt in de niet-representatieve ST-omgeving getest voor een beperkt aantal situaties.

Voorbeeld GAT

Kenmerk	RK MTP	RK MTP		Deelsys1	Deelsys2	Totaalsys
Functionaliteit	n.v.t.	n.v.t.		A/●●● functionele	B/-	C/● regressie
Gebr.vriendelijkheid	B	●●		C/I	-	B/●● usability
Beveiliging	A	●				
- autorisatiematrix	B			-/S autorisatietest	-/S autorisatietest	B/●● procestest
- appplicatie	C			-	-	C/● penetratietest
Inpasbaarheid	B	●●●		B/● scenariotest	C/● scenariotest	A/●●● procestest
…						

In bovenstaande voorbeeld wordt voor deelsysteem 1 opnieuw een functionele test uitgevoerd, gebruikmakend van een aantal van de testgevallen uit de ST maar in ATomgeving. Bij meerdere opleveringen vindt een regressietest op

het totale systeem plaats. Er vindt een usabilitytest in eigen omgeving plaats en in andere tests vindt een impliciete test op gebruikersvriendelijkheid plaats. De autorisatiematrix wordt getoetst op correcte invulling voor deelsystemen 1 en 2. Ook worden de aan deelsysteem 1 en 2 gerelateerde bedrijfs(deel)processen gesimuleerd door middel van het naspelen van gebruiksscenario's. Hierna wordt de werking van het totale systeem in combinatie met de bedrijfsprocessen getest. Een lichte penetratietest wordt ook voorzien.

Een eerste opzet van de strategie is vaak al te bereiken in de PRA-sessie, ofwel PRA en deze stap van de Strategiebepaling kunnen worden gecombineerd. Lukt dit niet, dan doet de testmanager een voorstel.

Een aandachtspunt is dat wanneer het MTP een zwaarte van ● ● ● voor een bepaalde testsoort (bijvoorbeeld ST) of een bepaalde combinatie van kenmerk/deelobject geeft. Dit niet wil zeggen dat in de ST het gehele systeem of de combinatie kenmerk/deelobject met de zwaarst mogelijke dekking getest wordt, maar dat er zwaarder getest moet worden dan gemiddeld. Dit moet ook blijken uit de toelichting bij het MTP.

Tips

- De teststrategie zal bij iteratieve of agile systeemontwikkeling met de vele tussen releases (iteraties, incrementen) veel aandacht moeten geven aan regressietesten. Ook moet de test(strategie) zich beperken tot de eigenschappen van de tussenrelease en niet een strategie formuleren alsof het de eindrelease betreft. Dit lijkt makkelijker dan het is, want in de PRA geven de gebruikers hun risicoinschatting op basis van de verwachte eindrelease.
- De testmanager moet bij het opstellen van de strategie al zoveel mogelijk rekening houden met de afweging tegen kosten, tijd en benodigde vaardigheden. Als hij weet dat er maar zeer beperkt budget is of dat de beschikbare mensen geen ervaring hebben met testen, moet het voorstellen van "onmogelijke" strategieën zoals heel dure tests of zeer grondige testontwerptechnieken voorkomen worden (om te veel terugkoppeling-slagen te vermijden)
- Het is aan te bevelen bij de keuze voor licht/zwaar testen voor een kenmerk/deelobject al gelijk een inventarisatie te maken van de te hanteren testbasis, omdat de beschikbaarheid en mate van detail van testbasis invloed kan hebben op de begroting en planning. In deze stappen kan daar dan rekening mee gehouden worden.

Uitgediept

Onderhoud

De foutkans is het voornaamste verschil tussen nieuwbouw en onderhoud. Het formuleren van de wijzigingen als deelobject vergemakkelijkt de strategie. Er is een aantal variaties mogelijk:

- een beperkte test, alleen gericht op de wijziging;
- een volledige (her-)test van de functie waarin de wijziging is aangebracht;
- het testen van de samenhang tussen de gewijzigde functie en de functies er direct omheen;
- het gehele systeem testen.

Regressietest

Tevens wordt de regressietest van het systeem als geheel onderkend. De regressietest richt zich voornamelijk op de samenhang tussen de gewijzigde en ongewijzigde delen van het systeem, omdat hier de kans op regressie het grootst is. Indien de PRA voor de nieuwbouw beschikbaar is, kunnen de hier toegekende risicoklassen aan kenmerken/deelobjecten een rol spelen bij de samenstelling van deze regressietest. Een regressietest kan beperkt of volledig worden uitgevoerd, afhankelijk van de risico's én van de benodigde testinspanning. Met behulp van de schaalbare regressietestset (zie paragraaf 6.6 "Fase Specificatie") is het heel makkelijk om een lichtere of zwaardere regressietest uit te voeren. Dit geeft veel flexibiliteit bij het testen van latere releases. Meer informatie, zie paragraaf 10.3 "Testvormen".

Producten

De teststrategie, vastgelegd in het testplan, met een korte beschrijving van de geplande testvormen en een indicatie van de zwaarte per kenmerk/deelobject.

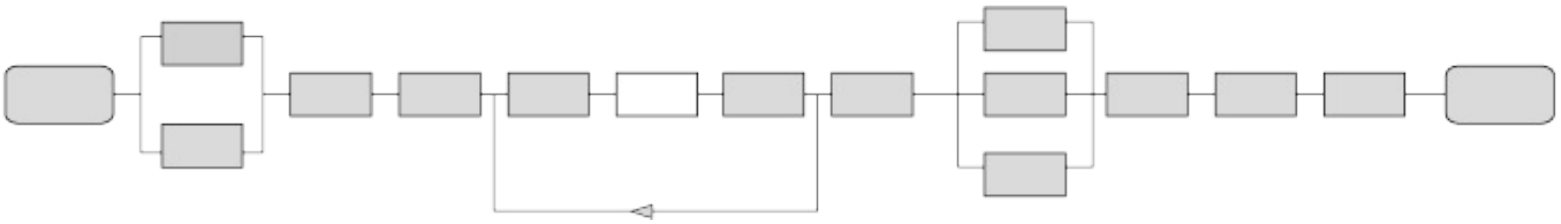
Technieken

Strategiebepaling (zoals beschreven in deze paragraaf).

Tools

Niet van toepassing.

6.2.6 Bepalen begroting



Doel

Het opstellen van een begroting voor de testsoort, gebaseerd op de teststrategie, zodat de opdrachtgever deze kan accepteren of bijstelling kan vragen.

Werkwijze

In het mastertestplan kan al een begroting voor de testsoort zijn opgesteld. Toch blijft deze stap nodig. De testmanager moet op basis van de opgestelde strategie bepalen hoeveel uren en eventueel geld nodig zijn. Wanneer dit buiten de marges van de toegekende begroting uit het mastertestplan valt, moet de testmanager dit afstemmen met de testmanager van het overkoepelende testproces. Daarna wordt de strategie of de begroting aangepast.

In de praktijk maakt het aantal benodigde testuren vrijwel altijd onderdeel uit van de begroting. Een ander, minder vanzelfsprekend onderdeel van de begroting is het financiële deel. Hoeveel kosten die uren? Is sprake van interne of externe inhuur of zelfs outsourcing, wat zijn de tarieven? Maar ook: hoeveel kosten de testomgeving, testtools en werkplekken? Wanneer de opdrachtgever hier behoefte aan heeft, moet de testmanager ook een financiële begroting opstellen.

Binnen een testsoort wordt vervolgens de benodigde tijd voor de verschillende fasen, zoals Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding vastgesteld. Bij de start van elke testfase begroot de testmanager de afzonderlijke testactiviteiten, zie hiervoor paragraaf 6.3.2 “Bewaken”.

Uitgediept

Begrotingstechnieken

De verschillende begrotingstechnieken en de stappen om tot een begroting te komen staan beschreven in hoofdstuk 11 “Begrotingstechnieken”. Voor een testplan kan worden begroot op basis van:

- verhoudingsgetallen
- testobjectomvang
- work breakdown structure (WBS)
- proportioneel begroten
- testpuntanalyse (TPA).

Om de betrouwbaarheid van de begroting te vergroten is het gebruik van zowel eigen ervaringscijfers als meerdere begrotingsmanieren en -technieken sterk aan te bevelen.

Tips

- Soms wordt voor het testen door de opdrachtgever een totaalbudget opgelegd.
  Deze moet verdeeld worden over de testfasen en de kenmerk/deelobject-combinaties. Enkele van de in hoofdstuk 11 beschreven technieken (met name bij Verhoudingsgetallen en Work Breakdown Structure) bieden hier hulp. Ook moet uitgezocht worden of het budget voldoende is. Er zijn de volgende tips te geven, maar veel komt aan op ervaring van de testmanager:
  1. Het beste is om een begroting te berekenen door de optelsom te maken van kenmerk/diepgang-begrotingen (bijvoorbeeld in PAT een lichte performancetest + een zware securitytest = 120 + 200 uur = 320 uur).
  2. Een andere optie is om het totaalbudget te toetsen door aan de hand van de globale vuistregels voor testsoorten de optelsom te maken: 15% van het totale projectbudget van 5000 uur voor de ST, 20% voor de AT = respectievelijk 750 en 1000 uur.
  3. Derde optie is om een standaardverdeelsleutel voor kenmerken te hanteren, die is vastgesteld op basis van een aantal ervaringen. Een voorbeeld is om Functionaliteit bij risicoklasse B standaard 70% van het totaalbudget te geven, bij A 80% en bij C 60%. Je kunt dit ook doen met een vast aantal uren, bijvoorbeeld gebruikersvriendelijkheid kan 70 uur krijgen bij klasse C tot 130 bij klasse A (voor een gemiddeld systeem). Hiermee kan beter de realiteitswaarde van de afzonderlijke begrotingen per testsoort/kenmerk beoordeeld worden.
  4. Vergelijk het toegekende budget met de budgetten én uiteindelijk bestede uren voor vergelijkbare trajecten in het verleden, zowel bij de organisatie zelf als mogelijk bij andere, gelijksoortige organisaties.

De begroting moet beoordeeld worden op realiteitswaarde en in de buurt van het toegekende totaalbudget zitten. Anders moeten er aanpassingen worden gedaan, door te kiezen voor een hoger totaalbudget, of minder kenmerken testen en/of minder diepgang van testen.

De hier gebruikte cijfers zijn realistische voorbeelden. Meer informatie is te vinden in hoofdstuk 11 “Begrotingstechnieken”.

- Met een diepgang van ● ● ● voor een kenmerk in een bepaalde testsoort (bijvoorbeeld Functionaliteit in de ST) wordt bedoeld dat het systeem zwaarder dan gemiddeld getest moet worden, niet dat het gehele systeem op de grondigst mogelijke manier getest moet worden. Zie ook de opmerking bij Teststrategie.
- Het opstellen van een begroting voor de test heeft een hoge onzekerheidsmarge.
  Het is belangrijk dat de testmanager aan de belanghebbenden duidelijk maakt dat de begroting gebaseerd is op een aantal aannames en daarom later mogelijk herzien moet worden. Een mogelijke oplossing is het gebruik van onzekerheidsmarges.
  In het begin van de test is de marge bijvoorbeeld ± 40%, bij start van de testuitvoering wordt dit ± 25%, ergens halverwege wordt het ± 10%.
- De koppeling tussen begroting en diepgang van testen (met bepaalde testtechnieken) is vrij ondoorzichtig.
  Hoeveel extra tijd kost bijvoorbeeld het toepassen van de elementaire vergelijkingentest ten opzichte van de datacombinatietest? Er zijn hier nog maar weinig ervaringscijfers voorhanden, veel gaat op basis van de ervaring en intuïtie van de testmanager.
- Ook hebben diverse andere factoren (kwaliteit van de testers, kwaliteit van het testobject en de testbasis, testomgeving, testtools) een (grote) invloed op de begroting. Deze factoren zijn óf nog niet bekend op het moment dat de begroting wordt opgesteld óf hun effect op de begroting is erg onduidelijk. De testmanager moet hier aannames voor doen, deze eventueel als uitgangspunt in het plan opnemen en in ieder geval de aannames toetsen zodra mogelijk.
- Het is moeilijk de benodigde onderhoudstestinspanning te verkopen aan management. Een algemeen “test imago” probleem is dat testen in de perceptie van management teveel kost. Bij testen in onderhoud wordt dat nog eens versterkt door het feit dat testen een relatief groot aandeel heeft in het onderhoudstraject, tot 80% aan toe. Dit heeft ermee te maken dat de totale testkosten bestaan uit *vaste en variabele kosten.* Denk bij vast aan bijvoorbeeld de inspanning die nodig is om de testomgeving klaar te zetten of aan het uitvoeren van een “standaard” regressietest en denk bij variabel aan bijvoorbeeld het voorbereiden en testen van doorgevoerde wijzigingen. Bij het testen van een kleine wijziging is het percentage vaste kosten hoog, omdat ongeacht de omvang van de wijziging altijd de omgeving moet worden klaargezet en de regressietest gedraaid. Naarmate de wijzigingen omvangrijker worden neemt het percentage vaste kosten af. Voorbeeld: bij het testen van een wijziging wordt altijd een 4 uur durende regressietest uitgevoerd. Als het testen van een wijziging in het totaal 8 uur duurt, bedraagt het percentage vaste testtijd 50% (4/8). Duurt het testen van één of meer wijzigingen in het totaal 40 uur, dan daalt dit percentage naar 10% (4/40).
  In het algemeen bevindt het aandeel testen (vast+variabel) zich tussen 35% - 80% van alle onderhoudsactiviteiten.

Het is aan de testmanager om dit duidelijk te maken aan de opdrachtgever en het (test)belang te bepleiten van grotere, beheerste releases, waarin veel wijzigingen gebundeld zijn, ten opzichte van constant kleine wijzigingen doorvoeren.

Producten

De begroting voor de testsoort, in uren en optioneel in geld, voorzien van hierbij gehanteerde aannames en uitgangspunten, vastgelegd in het testplan.

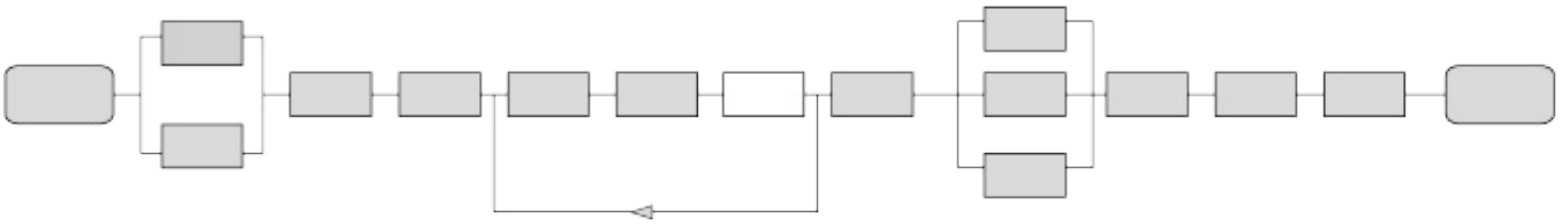
Technieken

Diverse begrotingstechnieken (hoofdstuk 11).
Stappenplan voor opstellen begroting (hoofdstuk 11).

Tools

Plannings- en voortgangsbewakingstool (een begrotingsspreadsheet is te vinden op www.tmap.net).

6.2.7 Bepalen planning



Doel

Het opstellen van een zo betrouwbaar mogelijke planning voor de testsoort, zodat de opdrachtgever hier rekening mee kan houden of kan bijsturen. Uitgangspunt van de planning is om de belangrijkste fouten (waarvan het vinden tot het beschouwingsgebied van de testsoort hoort) het eerst te vinden.

Werkwijze

Aan de hand van de planning van het systeemontwikkelproces en het mastertestplan wordt een planning voor de testsoort opgesteld. De testmanager geeft per fase de start- en einddatum en de op te leveren producten aan. De planning dient minimaal te omvatten:

- uit te voeren activiteiten (op activiteitniveau per fase);
- relaties met en afhankelijkheden van andere activiteiten (binnen of buiten de testsoort en tussen de diverse fasen en andere testsoorten);
- te besteden tijd per fase;
- benodigde en beschikbare resources (mensen en infrastructuur);
- benodigde en beschikbare doorlooptijd;
- op te leveren producten.

Afhankelijk van de behoefte van de opdrachtgever moeten de financiële gevolgen van de gemaakte keuzen zichtbaar worden gemaakt in een financiële planning. Denk hierbij aan het uitzetten van de kosten in de tijd voor het (intern en extern) personeel, training, werkplekken, testomgevingen en -tools.

Uitgediept

Bij het opstellen van een planning gelden de volgende uitgangspunten:

- De teststrategie en begroting vormen de basis voor het opstellen van de planning.
- In een goede planning worden de kenmerken/deelobjecten met hoog risico zo vroeg mogelijk getest.
- Bij een optimale testplanning worden zoveel mogelijk alleen de testuitvoeringsactiviteiten op het kritieke pad van het project uitgevoerd.
- Als er nog geen ervaringscijfers met betrekking tot het aantal hertesten voorhanden zijn, is het aan te bevelen om bij het maken van een planning rekening te houden met gemiddeld één hertest. Op basis van ervaringen uit het verleden kan besloten worden om meer of minder tijd voor hertesten in te plannen.
- Met name bij onderhoud en bij iteratieve of agile ontwikkelingtrajecten is het belang rijk om rekening te houden met het uitvoeren van regressietesten.
- Houd bij het maken van een planning rekening met benodigde tijd van derden. Denk hierbij bijvoorbeeld aan hersteltijd voor bevindingen of tijd voor het klaarzetten van de testomgeving.
- Het overdragen van het testobject naar en het installeren in de testomgeving valt in planningen vaak tussen wal en schip, of liever gezegd, tussen de planning van ontwikkeling en van testen in. Met name de eerste keren blijkt deze activiteit significant tijd te kosten, eerder in dagen dan in uren. Houd hier rekening mee.
- Probeer de in- en uitstroom van personeel zo geleidelijk mogelijk te laten verlopen waardoor er geen pieken en dalen in de bezetting voor komen.
- Meer planningsaanwijzingen zijn te vinden in de IT-klassieker [Brooks, 1975/1995].

Uitgediept

Benodigde informatie

Om uitgaande van een begroting een planning op te stellen, is aanvullende informatie nodig over de volgende onderwerpen:

- Beschikbare resources;
  Opmerking hierbij is dat bij het opstellen van een begroting maar beperkt rekening gehouden wordt met de beschikbare resources maar dat er een uitspraak gedaan wordt over het benodigde aantal uren. In combinatie met een deadline betekent dit dat er een bepaald aantal resources benodigd is om de geplande tests uit te voeren. In de praktijk is het vaak zo dat het beschikbare aantal resources in eerste instantie niet overeenkomt met het benodigde aantal resources. De testmanager moet dit expliciet maken en vervolgens bespreken met de opdrachtgever. Oplossingen kunnen zijn het inhuren van tijdelijk personeel, de doorlooptijd verlengen of de strategie aanpassen.
- Beschikbare doorlooptijd;
  In de praktijk is de beschikbare doorlooptijd veelal beschikbaar in de vorm van een deadline voor de betreffende fase.
- Beschikbaarheid van middelen zoals bijvoorbeeld testomgevingen en testtools;
  Wanneer zijn deze beschikbaar voor de activiteiten? Moeten bijvoorbeeld de testtools nog geselecteerd, aangeschaft en ingericht worden?
- Afhankelijkheden tussen de verschillende activiteiten;
  Activiteiten die afhankelijk zijn van andere activiteiten, kunnen pas starten na afronding van die andere activiteiten en niet parallel daaraan.
- Methode van systeemontwikkeling;
  Afhankelijk van de manier waarop het systeem wordt ontwikkeld, kunnen de testsoorten worden ingepland. Bij een watervalmethode is sprake van een andere fasering dan in een iteratief traject waarbij test- en ontwikkelactiviteiten parallel en soms geïntegreerd worden uitgevoerd. De ontwikkeltest en systeemtest hebben hier als regel meer mee te maken dan de acceptatietest.
- Informatie over mijlpaalmomenten van het ontwikkelproject.
  Deze informatie is nodig om de testplanning optimaal te laten aansluiten bij de planning van de rest van het project. Hierdoor is het mogelijk om de totale doorlooptijd van het project te minimaliseren.

De planning wordt weergegeven in bijvoorbeeld een netwerkplanning of een balkenschema, afhankelijk van de binnen de organisatie gehanteerde techniek. In dit boek worden geen planningstechnieken behandeld. Reden hiervoor is dat de testmanager voor het plannen van het testtraject gebruik maakt van standaard planningstechnieken die niet specifiek zijn voor het testproces.

Voorbeeld Activiteitenplanning

Testfase	Weeknummer (2006)													Uren/FTE
	14	…	20	21	…	34	35	36	37	38	39	40	41	42	
Planning, Beheer en Inrichting & beheer infrastructuur	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**2 FTE**
Voorbereiding en Specificatie	X	X	X	X	X	X	X								
Uitv. FAT, FIT (functionele integratietest)								X	X	X	X				**3480 uren**
Uitv. KIT (keten-integratietest)										X	X	X	X		**3480 uren**
Uitv. GAT										X	X	X	X		
Afronding														X	
Reserveweek														X	
														Totaal:	**2 FTE + 6960 uren**

Voorbeeld Mijlpalenplanning

Mijlpaal	Datum	Verantwoordelijke
Opleveren definitieve testbasis	01-03-2006	Projectleider SAP
Opleveren testinfrastructuur	31-08-2006	Projectleider SAP
Opleveren testobject	31-08-2006	Projectleider SAP
Afronden testspecificaties FAT, FIT, KIT, GAT	31-08-2006	Testcoördinator
Afronden testuitvoering	14-10-2006	Testcoördinator
Opleveren testware	22-10-2006	Testcoördinator
Opleveren Voorlopig Vrijgaveadvies	15-10-2006	Testmanager
Opleveren Vrijgaveadvies	22-10-2006	Testmanager
Opleveren Testrapport	22-10-2006	Testmanager

Tip

Geef bij de planning van resources al aan vanaf welk moment het niet meer mogelijk is om dreigende uitloop op te vangen door extra mensen in te zetten. Soms is een omgeving zo complex of specifiek dat “een mannetje erbij” geen tijdswinst meer oplevert. Het is niet prettig dit nog uit te moeten leggen wanneer het moment al daar is dat de projectleider bezig is “extra handjes” voor het testteam te regelen en al helemaal niet wanneer die “extra handjes” al aan het team voorgesteld worden…

Een aan kwaliteit gerelateerd aspect van planning is wanneer een testsoort klaar is en het testobject kan overdragen aan de volgende testsoort of aan productie. Met andere woorden, wat mag de ‘volgende’ testsoort verwachten nadat de ‘voorgaande’ testsoort is afgerond. Om deze verwachtingen expliciet te maken, worden eisen aan het resultaat van de testsoort gesteld. In de praktijk worden deze eisen ook wel exitcriteria genoemd. Met het toenemen van outsourcing wordt het steeds belangrijker heldere exitcriteria vast te stellen om te voorkomen dat de leverancier onvoldoende kwaliteit levert.

Uitgediept

Exitcriteria kunnen bijvoorbeeld gerelateerd worden aan het aantal bevindingen van een bepaalde ernstcategorie dat nog open mag staan, de manier waarop een bepaald risico is afgedekt (bijvoorbeeld alle systeemdelen met de hoogste risicoklasse zijn met een formele testontwerptechniek getest), of de mate waarin de requirements moeten zijn getest.

Vanuit het mastertestplan worden exitcriteria aan de testsoort opgelegd. Als dat niet het geval is of er geen mastertestplan is, moet de testmanager de criteria overeenkomen met de opdrachtgever.

In het onderstaande kader staat een aantal concrete voorbeelden van exitcriteria:

Systeem X mag pas aan de acceptatietest worden overgedragen als aan de volgende voorwaarden wordt voldaan:
- er staan geen bevindingen meer open van categorie “ernstig”
- er staan maximaal 4 bevindingen open van categorie “storend”
- er staan in totaal maximaal 20 bevindingen open
- voor alle openstaande bevindingen is een workaround beschreven
- voor alle gebruikersfunctionaliteit zijn minimaal de goed-paden getest en akkoord bevonden

Systeem X mag aan de AT worden overgedragen als schriftelijk aangetoond kan worden dat alle risico’s die conform document Y aan de ST zijn toegewezen met de afgesproken diepgang en testwerkwijze zijn getest.

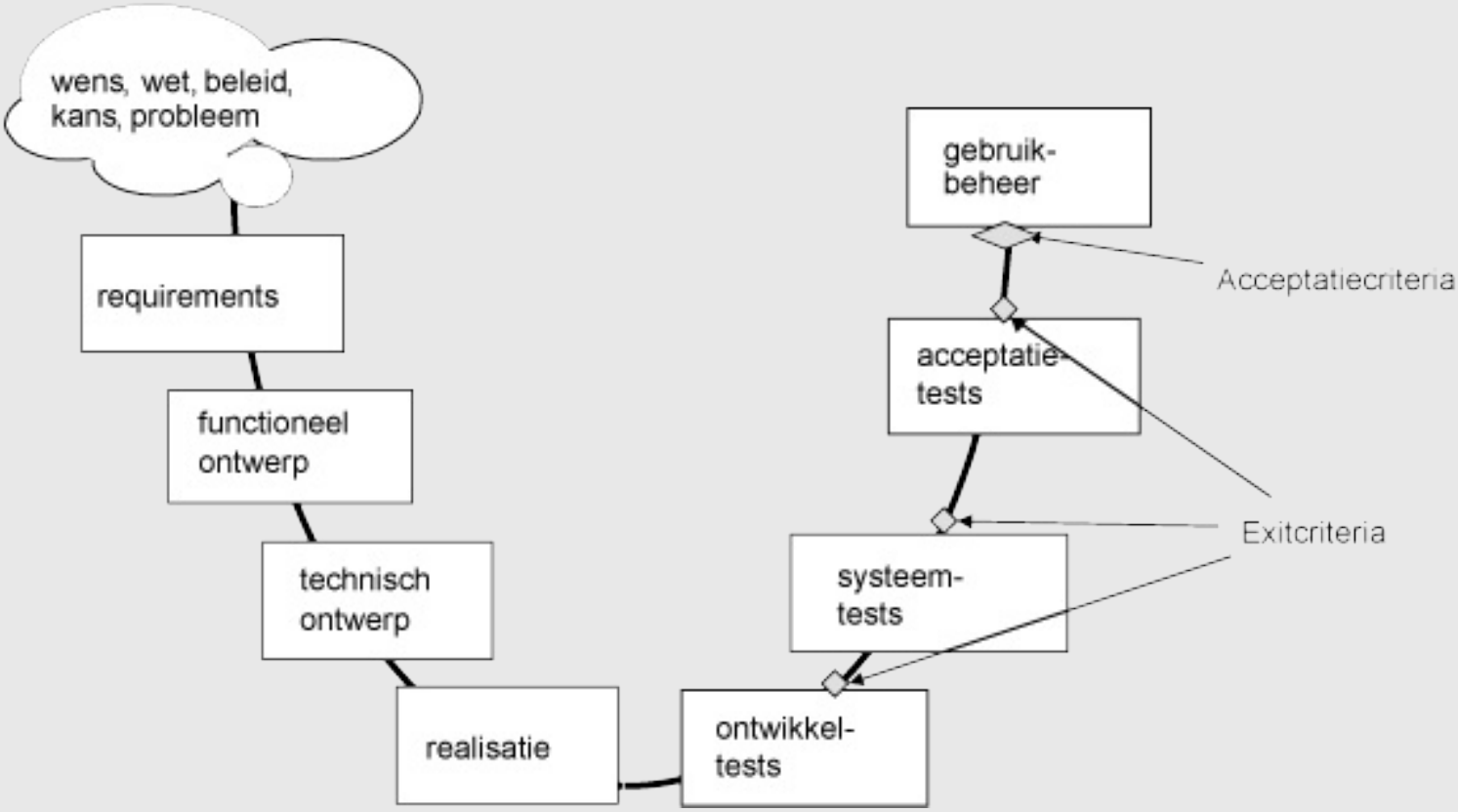
Belangrijk aandachtspunt bij de bovenstaande criteria is dat er door alle betrokkenen eenduidige definities

afgesproken moeten worden over wat een bepaalde ernstcategorie is en wat met ‘afgesproken diepgang en testwerkwijze’ wordt bedoeld. In de praktijk kan onduidelijkheid hierover tot heftige discussies leiden.

Uitgediept

Overeenkomst en verschil acceptatie- en exitcriteria

Als andere term voor exitcriteria wordt ook wel acceptatiecriteria gebruikt, zoals besproken in paragraaf 6.2.2. Naast het feit dat acceptatiecriteria breder *kunnen* zijn dan exitcriteria, is een ander verschil dat acceptatiecriteria aan het eind komen, bij acceptatie dus, en exitcriteria bij de overgang van een testsoort naar de andere testsoort of naar productie. Figuur 6.6 maakt dit duidelijk.



Figuur 6.6: Exit- en acceptatiecriteria.

Tip

Suspend- en resume-criteria

In sommige, met name formeel geregelde, tests kunnen ook zogenaamde suspend en resume criteria in het plan gedefinieerd worden. Deze criteria geven aan onder welke omstandigheden het testen tijdelijk stopt (suspend) en daarna weer doorgang vindt (resume). Voorbeelden van suspend-criteria zijn dat testen moet stoppen wanneer een bepaalde infrastructuur-component niet beschikbaar is of een testblokkerende bevinding gedaan wordt. Een resume-criterium kan zijn dat bij opheffing van het suspend-criterium de test van deelsysteem/functie/component geheel opnieuw moet plaatsvinden.

Terugkoppeling

Wanneer de testmanager een planning heeft opgesteld is dit het moment om af te stemmen met de opdrachtgever. Zijn de opgestelde teststrategie en hieruit volgende begroting en planning niet acceptabel, dan herhalen deze stappen zich. Hierbij maken de opdrachtgever en testmanager de afweging om bepaalde aspecten minder zwaar te testen, zodat tijd en/of geld bespaard wordt, maar meer risico gelopen wordt, of juist andersom. Om de communicatie te vergemakkelijken grijpt de testmanager hierbij weer terug naar de oorspronkelijke testdoelen.

Wanneer een mastertestplan is opgesteld, wordt de overkoepelende testmanager hierbij wel betrokken, maar maakt de opdrachtgever de uiteindelijke keuze.

Een aangepaste strategie ziet er als volgt uit, waarbij minder testzwaarte wordt getoond door ○ in plaats van ● en meer testzwaarte door ●̲ .

Voorbeeld ST

Kenmerk	RK MTP	RK MTP		Deelsys1	Deelsys2	Totaalsys
Functionaliteit	n.v.t.	n.v.t.		A/●●○ functionele, regressie	B/●○ functionele, regressie	C/● integratie, multi-user
Performance online	B	●		-	-	C/● steekproef in ST-omgeving
…						

Voorbeeld GAT

Kenmerk	RK MTP	RK MTP		Deelsys1	Deelsys2	Totaalsys
Functionaliteit	n.v.t.	n.v.t.		A/●○ functionele	B/-	C/● regressie
Gebr.vriendelijkheid	B	●●		C/I	-	B/●○ usability
Beveiliging	A	●				
- autorisatiematrix	B			-/S autorisatie-test	-/S autorisatie-test	B/●● procestest
- appplicatie	C			-	-	C/● penetratietest
Inpasbaarheid	B	●●●		B/●●̲ scenariotest	C/● scenariotest	A/●●○ procestest
…						

De aangepaste strategie leidt tot een andere begroting en planning, maar ook in een indicatie van grotere (of juist kleinere) productrisico's, vertaald in voor de opdrachtgever begrijpelijke termen (teruggrijpend op de productrisicoanalyse met testdoelen, kenmerken en deelobjecten).

In aanvulling op de terugkoppeling van strategie, begroting en planning bespreekt de testmanager met opdrachtgever het hanteren van toleranties bij het uitvoeren van het testproces. Dit zijn grenzen waarbinnen de testmanager de opdrachtgever geen toestemming hoeft te vragen. Zo wordt vaak een tolerantie van 5% voor de begroting aangehouden. Voor de planning kan afgesproken worden dat alleen afwijkingen van projectmijlpalen besproken moeten worden. Bij strategietoleranties is bijvoorbeeld voor het één niveau lichter of zwaarder testen van een kenmerk/deelobject geen toestemming vooraf van opdrachtgever nodig.

Producten

Planning voor het testproces.
Exitcriteria.
Optioneel: toleranties voor strategie, begroting en planning
Optioneel: suspend en resume criteria (bovenstaande producten worden vastgelegd in het testplan).
Met de opdrachtgever teruggekoppelde strategie, begroting en planning.

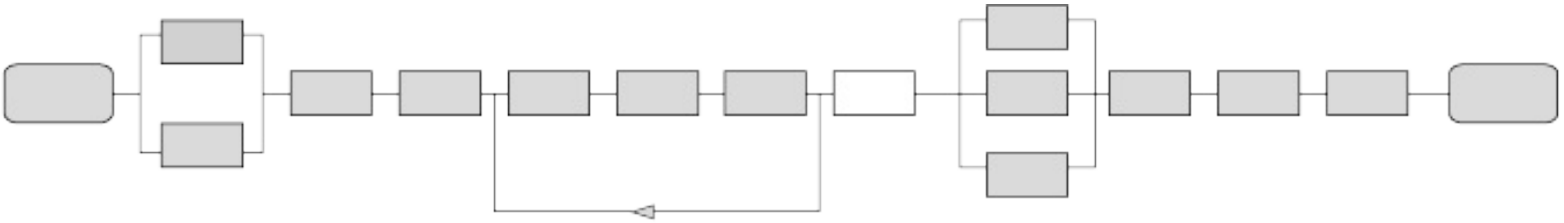
Technieken

Niet van toepassing.

Tools

Plannings- en voortgangsbewakingstool.

6.2.8 Toewijzen testeenheden en testtechnieken



Doel

Het toewijzen testeenheden en testtechnieken heeft als doel het concretiseren en beheersbaar maken van de testvormen en het licht/zwaar testen van kenmerken/deelobjecten op basis van de goedgekeurde teststrategie, begroting en planning.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen testeenheden;
2. Toewijzen testtechnieken.

Voor deze stap is informatie nodig die er in de praktijk nog niet altijd is. In dat geval voert de testmanager deze stap op globale wijze uit en brengt de details in een later stadium, tijdens de fase Beheer, aan.

1. Bepalen testeenheden

In de strategie zijn aan de kenmerken/deelobjecten testvormen en een testzwaarte toegekend. In sommige gevallen kan een testvorm voor een bepaald kenmerk/deelobject heel omvangrijk zijn. Om hiervoor beheersbare en uitvoerbare activiteiten te kunnen definiëren splitst de testmanager het deelobject verder op in zogenaamde testeenheden.

Definitie

Een testeenheid is een verzameling processen, transacties en/of functies die gezamenlijk worden getest.

Het voordeel van een testeenheid is dat het een beheersbare eenheid werk vormt (X uur in periode Y) en als zodanig een belangrijk stuurmechanisme voor de testmanager is. Redenen om een deelobject te splitsen in testeenheden zijn:

- de omvang van het deelobject is te groot om het testen ervan straks goed te kunnen beheersen;
- een bepaald onderdeel van het deelobject vergt een aparte testaanpak met andere testtechnieken, bijvoorbeeld omdat het risico sterk afwijkt of omdat de aard van het onderdeel afwijkt van de rest (scherm versus verwerking).

Omdat een testeenheid een eenheid werk voorstelt, doet de testmanager er verstandig aan dit af te stemmen met de ontwikkelaar, zodat een oplevereenheid overeenkomt met één of meer testeenheden en er geen halve testeenheden worden opgeleverd.

2. Toewijzen testtechnieken

Een volgende stap is dat per testvorm op basis van de gekozen zwaarte één of meer passende testtechnieken worden gekozen waarmee de test gespecificeerd en uitgevoerd moet worden. Indien een deelobject in testeenheden is verdeeld, worden technieken per testeenheid toegewezen.

Maar hoe nu de passende technieken te kiezen? Hoofdstuk 14 “Testontwerptechnieken” bevat een groot aantal testontwerptechnieken, hierop kunnen variaties gemaakt worden, er zijn andere technieken, inclusief zelf bedachte technieken en ook checklists kunnen als techniek gebruikt worden. Deze keuze is, naast gekozen zwaarte, sterk

afhankelijk van een aantal andere aspecten:

- Testbasis
  Moeten de tests gebaseerd worden op requirements, is het functioneel ontwerp in pseudo-code opgeschreven of daar makkelijk toe te herleiden, zijn er state-transition diagrammen of beslissingstabellen of is het zeer informeel en zit veel kennis in de hoofden van de materiedeskundigen? Sommige technieken leunen zwaar op het beschikbaar zijn van een bepaalde vorm van beschreven testbasis, bij andere technieken kan de testbasis een ongestructureerde en slecht gedocumenteerde verzameling informatiebronnen zijn.
- Testvorm / kwaliteitsattributen
  Wat moet er getest worden? Sommige testontwerptechnieken zijn met name geschikt voor het testen van de interactie (schermen, rapporten, online) tussen systeem en gebruiker, andere zijn meer geschikt voor het testen van de relatie tussen de administratieve organisatie en het systeem, voor het testen van performance of beveiliging, of voor het testen van complexe verwerking (berekeningen) en weer andere zijn bedoeld om de integratie tussen functies en/of gegevens te testen. Ook worden vaak checklists gebruikt voor het testen van niet-functionele kwaliteitsattributen. Dit alles heeft een relatie met het soort fouten dat met behulp van de techniek gevonden kan worden, bijvoorbeeld foutieve invoercontroles, onjuiste verwerking of integratiefouten.
- Wat voor soort variaties moeten hoe zwaar gedekt worden?
  Hoe zwaar moet er getest worden? Dit moet uitmonden in het definiëren van één of meer dekkingsvormen en daaraan gerelateerde basistechnieken, zie ook paragraaf 14.3 “Dekkingsvormen en basistechnieken”.
- Kennis en kunde van de beschikbare testers.
  Zijn de testers al getraind in de techniek, hebben ze er al ervaring mee of betekent de keuze voor een bepaalde techniek dat de testers hierin opgeleid en gecoacht moeten worden? Is de techniek eigenlijk wel geschikt voor de beschikbare testers? Gebruikers zijn meestal geen professionele testers.
- Arbeidsintensiviteit
  Hoe arbeidsintensief zijn de gekozen technieken en staat dit in verhouding tot de begrote hoeveelheid tijd? Soms moeten andere technieken met mogelijk wat andere dekking gekozen worden om binnen de begroting te blijven. Wanneer dit betekent dat er lichter getest gaat worden dan afgesproken is, moet dit natuurlijk wel teruggekoppeld worden met de opdrachtgever!

Na bovengenoemde aspecten meegenomen te hebben, maakt de testmanager een keuze voor de te hanteren technieken. Dit ziet er bijvoorbeeld als volgt uit:

Voorbeeld ST

Kenmerk	Deelobject	Testvorm		Technieken
Functionaliteit	Deelsys1 (A/● ●)	Functionele test		te1: DCT te2: SYN, SEM
Functionaliteit	Deelsys1 (A/● ●)	Regressietest		te3: selectie uit te1 en te2
Functionaliteit	Deelsys2 (B/●)	Functionele test		te4: Exploratory Testing te5: SYN, SEM
Functionaliteit	Deelsys2 (B/●)	Regressietest		te6: selectie uit te4 en te5
Functionaliteit	Totale systeem (C/●)	Integratie		te7: GCT
Functionaliteit	Totale systeem (C/●)	Multi-user		te8: Exploratory Testing
Performance on-line	Totale systeem (C/●)	Steekproef in ST-omgeving		te9: Error Guessing
…				

In dit voorbeeld is het testen van deelsysteem 1 verdeeld over testeenheden (“te”) 1 en 2, deelsysteem 2 bestaat uit testeenheden 4 en 5. Testeenheid 1 met veel complexe berekeningen krijgt met de DataCombinatieTest toch een vrij lichte techniek (omdat de opdrachtgever koos voor een gemiddelde zwaarte), testeenheid 4 bevat verwerkingsfunctionaliteit en krijgt de (heel vrije) “techniek” Exploratory Testing toegewezen, testeenheden 2 en 5 bestaan voornamelijk uit schermen en krijgen elk 2 technieken, de Syntactische en Semantische Test. Het totale systeem wordt vervolgens op samenhang getest met de GegevensCyclusTest (testeenheid 7) en het multi-user aspect met Exploratory Testing (testeenheid 8). Latere regressietests bestaan uit een selectie van al gemaakte testgevallen (testeenheden 3 en 6). Performance wordt tenslotte licht getest met behulp van Error Guessing (testeenheid 9).

Voorbeeld GAT

Kenmerk	Deelobject	Testvorm		Technieken
Functionaliteit	Deelsys1 (A/●)	Functionele test		te1: steekproef ST te2: steekproef ST
Functionaliteit		Regressie		te3: DCT

	Totale systeem (C/●)			
Gebr.vriendelijkheid	Deelsys1 (C/I)	Gebruikersvriendelijkheid		impliciet in te1 en te2
Gebr.vriendelijkheid	Totale systeem (B/●)	Usability		te4: SUMI
Beveiliging – aut.matrix	Deelsys1 (-/S)	Autorisatietest		te5: steekproef aut.tabel
Beveiliging – aut.matrix	Deelsys2 (-/S)	Autorisatietest		te5: steekproef aut.tabel
Beveiliging – aut.matrix	Totale systeem (B/● ●)	Procestest		te7: SEM
Beveiliging – applicatie	Totale systeem (C/●)	Penetratietest		te8: Error Guessing
Inpasbaarheid	Deelsys1 (B/● ●)	Scenariotest		te9: PCT, testmaat 2
Inpasbaarheid	Deelsys2 (C/●)	Scenariotest		te10: PCT, testmaat 1
Inpasbaarheid	Totale systeem (A/● ●)	Procestest		te11: PCT, testmaat 2

In dit voorbeeld wordt in testeenheden 1 en 2 gebruik gemaakt van ST-testgevallen. De regressietest op het totale systeem gebeurt met de lichte DataCombinatieTest.

Gebruikersvriendelijkheid wordt impliciet mee getest bij testeenheden 1 en 2 door na afloop de indrukken van de testers te evalueren. Daarna vindt een expliciete test plaats met behulp van de SUMI-checklist (zie ook paragraaf 10.3.2 “Usability”). De autorisatiematrix wordt eerst steekproefsgewijs gecontroleerd op juiste vulling, daarna worden de autorisaties met behulp van de semantische testtechniek dynamisch getest. De lichte penetratietest gebeurt met error guessing, Inpasbaarheid wordt met de procescyclustest getest.

Is de keuze gemaakt om dynamisch expliciet te testen, dan kan de onderstaande tabel hulp bieden bij het kiezen van de te hanteren testontwerptechnieken. De tabel geeft per kwaliteitsattribuut verschillende testontwerptechnieken die geschikt zijn voor het testen van dat attribuut. Deze tabel is ook op www.tmap.net te vinden.

Voor de relevante kwaliteitsattributen worden bruikbare testontwerptechnieken genoemd, waarbij een onderscheid is gemaakt ten aanzien van de dekking van de test. ● betekent een lichte dekking, ● ● een gemiddelde dekking en ● ● ● een zware dekking. De genoemde technieken moeten gezien worden als voor de hand liggende keuzes en zijn ter inspiratie bedoeld. De tabel is zeker niet bedoeld als voorschriftgevend, andere techniekkeuzes zijn zonder meer toegestaan.

Kwaliteitsattribuut	**Testontwerptechniek**		
	● / licht	**● ● / gemiddeld**	**● ● ● / zwaar**
Beheerbaarheid • installeerbaarheid	CKL	DCT	DCT
Beveiliging	CKL	SEM	Penetratietest
Bruikbaarheid	UCT	UCT PCT*	RLT
Continuïteit		RLT	RLT
Functionaliteit • overkoepelend	DCT	DCT GCT PCT*	DCT
Functionaliteit • detail	DCT	DCT EVT	DCT + grenswaarde EVT + grenswaarde BTT
Functionaliteit • validaties	SYN	SYN SEM	
Gebruiksvriendelijkheid	SYN	SYN UCT* PCT*	Usabilitytest (eventueel in lab)
Infrastructuur (geschiktheid voor)		RLT*	
Inpasbaarheid	PCT testmaat-1 UCT*	PCT	PCT testmaat-3
Performance		RLT	
Portabiliteit	CKL Steekproef functionele tests Steekproef omgevingscombinaties	Functionele regressietest Belangrijke omgevingscombinaties	Alle functionele tests Alle omgevingscombinaties
Zuinigheid		RLT	

Toelichting bij de bovenstaande tabel:

Gebruikte afkortingen:

* Wanneer de techniek enigszins wordt aangepast, is deze bruikbaar voor het testen van betreffende kwaliteitsattribuut

BTT	Beslistabeltest
CKL	Checklist
DCT	Datacombinatietest
EVT	Elementaire vergelijkingentest
GCT	Gegevenscyclustest
PCT	Procescyclustest (testmaat=2)
RLT	Real life test
SEM	Semantische test
SYN	Syntactische test
UCT	Use case test

Voor een uitgebreide beschrijving van deze technieken wordt verwezen naar hoofdstuk 14 “Testontwerptechnieken”.

Gebruikte begrippen:

Omgevingscombinaties

Bij testen van portabiliteit wordt gekeken of het systeem draait in diverse omgevingen. Omgevingen kunnen bestaan uit diverse zaken, zoals hardwareplatform, databasesysteem, netwerk, browser en operating system. Als het systeem moet kunnen draaien op 3 (versies van) operating systemen, onder 4 browser(versie)s, geeft dit al 3 x 4 = 12 te testen *omgevingscombinaties*.

Penetratietest

De penetratietest is gericht op het vinden van gaten in de beveiliging van het systeem. Deze test wordt meestal door een zogenaamde ethical hacker uitgevoerd.

Portabiliteit – functionele tests

Om portabiliteit te testen kan in een bepaalde omgeving, oplopend in zwaarte, een steekproef van de functionele tests worden uitgevoerd, de regressietest of alle testgevallen.

Usabilitytest

Een test waarin de gebruikers bedrijfsprocessen kunnen simuleren en het systeem kunnen beproeven. Door de gebruikers tijdens de test waar te nemen, worden uitspraken over de kwaliteit van het testobject gedaan. Een specifiek hiervoor ingerichte en gecontroleerde omgeving met onder andere videocamera’s en een kamer met spiegelglas voor de waarnemers wordt ook wel usabilitylab (“laboratorium”) genoemd.

Zie ook paragraaf 10.3 “Testvormen”.

Uitgediept

Testontwerptechnieken zijn eigenlijk eerst aan testvormen gekoppeld en via deze pas aan kwaliteitsattributen. Omdat er geen eenduidige set aan testvormen is, is gekozen voor een rechtstreekse koppeling aan kwaliteitsattributen.

De technieken Exploratory Testing en Error Guessing komen niet voor in de bovenstaande tabel. Reden hiervoor is dat deze technieken tot bevindingen kunnen leiden voor alle kwaliteitsattributen. Exploratory Testing is elke vorm van testen waarbij de tester zijn testontwerp maakt tijdens de testuitvoering. De informatie die wordt verkregen tijdens het testen wordt gebruikt om nieuwe en betere testgevallen te ontwerpen. Error Guessing houdt in dat testers zonder het gebruik van gedocumenteerde testgevallen het systeem ongestructureerd testen. Zie voor een beschrijving van Exploratory Testing en Error Guessing paragraaf 14.4 “Een basisset testontwerptechnieken”. Aangezien het niet mogelijk is om alle mogelijke testsituaties in testgevallen vast te leggen, zijn Exploratory Testing en Error Guessing waardevolle technieken om aanvullende testen uit te voeren. Advies is dan ook om tijdens iedere testperiode een beperkte hoeveelheid tijd in te ruimen voor deze technieken.

Is een cel leeg gelaten en zijn er geen voorliggende technieken genoemd, dan kunnen error guessing of exploratory testing toegepast worden. Zijn er voorliggende technieken benoemd, zoals bij Functionaliteit-validaties met zware dekking, dan kunnen voorgaande technieken als basis gebruikt worden. Vaak zijn deze technieken in een zwaardere variant uit te voeren of kunnen meerdere technieken gekozen worden.

Tip

Veel onzekerheid

In sommige gevallen is sprake van veel onzekerheid waar de risico's liggen. Het is dan moeilijk om een goede strategie te bepalen en de juiste technieken te kiezen.

Hiervoor zijn twee oplossingen mogelijk:

- Exploratory testing, omdat dit de flexibiliteit heeft om bij testuitvoering steeds in te zoomen op waar de risicogebieden blijken te zijn;
- Hanteren van het "ui"-model. Hierbij worden van tevoren vrij globale tests gespecificeerd maar wordt tijd en budget ingepland om tijdens testuitvoering, wanneer de risicogebieden duidelijker worden, aanvullende en diepgaandere tests te maken die hierop gericht zijn. De test gaat dus als het ware elke keer een laagje dieper.

Producten

Indeling in testeenheden met toewijzing van testtechnieken.

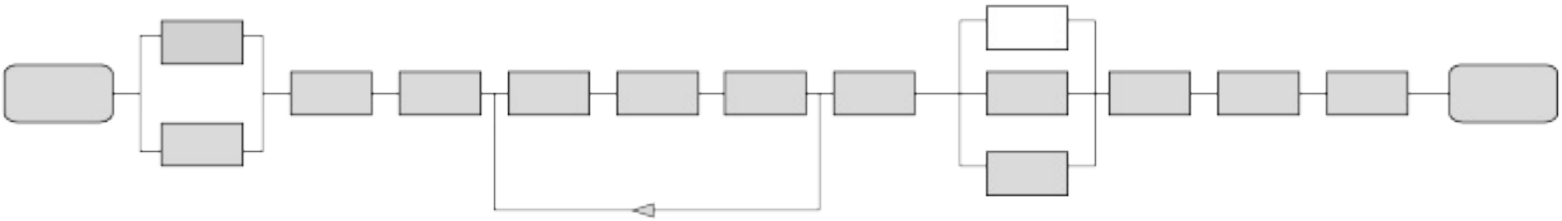
Technieken

Niet van toepassing.

Tools

Workflowtool.

6.2.9 Definiëren testproducten



Doel

Het eenduidig definiëren van de op te leveren testproducten.

Werkwijze

De activiteiten die uitgevoerd worden voor het plannen, uitvoeren en beheersen van het testproces leveren bepaalde producten op zoals het testplan en rapportages, testgevallen en -scripts, maar ook procedures, voorschriften en projectdocumentatie als overlegnotulen. In overleg met opdrachtgever en andere betrokkenen wordt bepaald wat de op te leveren producten zijn. Als er een mastertestplan is, zijn vanuit dit plan ook op te leveren testproducten gedefinieerd. Het kan gaan om testware zoals testplannen of testscripts of geautomatiseerde regressietests, dus producten die voor hergebruik in aanmerking komen, maar ook om testdocumentatie als voortgangsrapportages.

Tip

Het gebruik van tools voor configuratie management of planning en voortgangsbewaking helpt om een uniforme werkwijze af te dwingen.

De volgende testproducten worden onderkend:

- Testware

Definitie

Testware is alle testdocumentatie die tijdens het testproces wordt geproduceerd en voor onderhoudsdoeleinden gebruikt kan worden en daarom overdraagbaar en onderhoudbaar moet zijn.

Bij regressietesten wordt vaak gebruik gemaakt van bestaande testware. Testware omvat bijvoorbeeld:

- Testplan(nen)
  Omvat zowel mastertestplannen als andere testplannen;
- Logische testspecificaties
  De logische specificaties omvatten de logische beschrijving van de testgevallen;
- Fysieke testspecificaties
  De fysieke testspecificaties omvatten de fysieke beschrijving van de testgevallen en de testscripts. Fysiek houdt in dat de testgevallen uitvoerbaar en controleerbaar zijn. De fysieke testgevallen zijn getransformeerd uit de logische testgevallen en samengevoegd in de testscripts in de meest efficiënte volgorde van uitvoering;
- Traceerbaarheidsmatrix (of cross-reference-matrix)
  Een matrix waarin de link aangegeven is tussen de testbasis (requirements, functionele specificaties, enzovoort) en de daadwerkelijke testgevallen. De te testen situaties uit de testbasis staan verticaal en de testgevallen horizontaal. Zie ook Traceerbaarheid in paragraaf 6.2.12 “Inrichten beheer”;
- Testinvoerbestanden
  In de, op basis van de testscripts aangemaakte, testinvoerbestanden moet het volgende (kort) beschreven worden:
  – doel
  – de “fysieke” naam
  – aanmaakdatum
  – korte omschrijving van de inhoud
  – het soort bestand en andere relevante kenmerken
  – verwijzing naar de testscripts;
- Basisdocumentatie
  Een beschrijving van de testomgeving, testtools, testorganisatie en uitgangsdatabases;
- Testuitvoeringsdossier
  Het testuitvoeringsdossier bestaat uit:
  – testresultaten (logging van uitgevoerde tests en testgevallen) en rapportages
  – testuitvoer (optioneel)
    Het “bewijsmateriaal” van de uitgevoerde tests kan bestaan uit screendumps, printuitvoer en uitvoerbestanden. De tester levert na afronding van de test de geproduceerde uitvoer aan de beheerder op. De op te leveren testdocumentatie van de uitvoer bevat:
    – verwijzing naar de “fysieke” naam
    – aanmaakdatum
    – korte omschrijving van de inhoud
    – soort bestand en andere relevante kenmerken
    – verwijzing naar het testscript;
  – informatie over de bevindingen en de wijzigingen
  – overdracht en versiedocumentatie.
- Overige test(project)documentatie
  Tijdens het testproces worden diverse documenten ontvangen of zelf opgesteld, die niet voor hergebruik bedoeld zijn, zoals:
  – projectplannen;
  – verslagen van de overleggen (met besluiten- en activiteitenlijsten) ;

– correspondentie, zowel op papier als elektronisch (e-mail);
– memo's;
– (project)standaards en richtlijnen;
– test-, review- en auditrapporten;
– rapportages over voortgang en kwaliteit;
– enzovoort.

Door middel van een korte beschrijving worden de inhoud en het doel van de diverse producten en documenten aangegeven. Naast het benoemen van op te leveren producten kunnen ook normen en standaards gegeven worden en kan verwezen worden naar te gebruiken sjablonen.

Producten

Een beschrijving van de op te leveren testproducten inclusief normen en standaards en eventueel verwijzingen naar sjablonen, vastgelegd in het testplan.

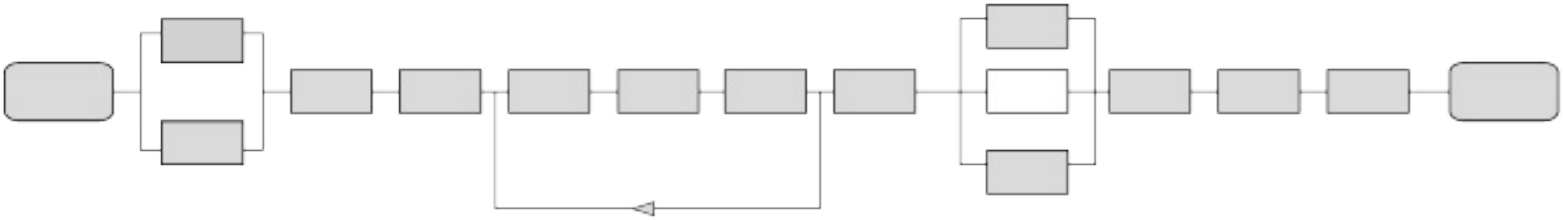
Technieken

Niet van toepassing.

Tools

Testwarebeheertool.

6.2.10 Definiëren organisatie



Doel

Het definiëren van de rollen, taken, bevoegdheden en verantwoordelijkheden die van toepassing zijn voor de testsoort.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Bepalen benodigde rollen;
2. Toewijzen taken, bevoegdheden en verantwoordelijkheden;
3. Beschrijven organisatie;
4. Toewijzen personeel;
5. Vaststellen opleidingen en coachingsbehoefte;
6. Vaststellen overleg- en rapportagestructuren.

1. Bepalen benodigde rollen

Om de activiteiten in het testproces goed te laten verlopen bepaalt de testmanager welke testrollen (zie paragraaf 8.6 "Testprofessionals" en hoofdstuk 16 "Testrollen") hiervoor moeten worden onderkend en ingevuld. Denk hierbij met name aan:

• Testmanagement of testcoördinatie;

- Testteamleiding;
- Tester;
- Beheer (testproces, testproducten, bevindingen);
- Intermediair;
- Ondersteuning (materiedeskundigheid, systeemkennis, testomgeving, testtools of testmethodisch).

Maak hierbij zoveel mogelijk gebruik van de op overkoepelend niveau ingerichte rollen.

2. Toewijzen taken, bevoegdheden en verantwoordelijkheden

Hier worden per benodigde rol de taken en verantwoordelijkheden weergegeven.

Uitgediept

Voorbeelden van taken

(met tussen haakjes de meest voor de hand liggende rol):

- het opstellen en onderhouden van het testplan; (testmanager)
- het doen uitvoeren, bewaken en bijsturen van de testactiviteiten; (testmanager)
- het uitvoeren van een detailintake op de testbasis; (tester)
- het op basis van gebruikersinformatie ontwerpen van tests; (tester)
- het specificeren van testgevallen en -scripts; (tester)
- het uitvoeren van tests; (tester)
- het inrichten van geautomatiseerde testuitvoering; (testtoolspecialist, testtoolprogrammeur)
- het laten inrichten van de technische infrastructuur en het laten beheren ervan; (testinfrastructuurcoördinator)
- het inrichten van methodische, technische en functionele ondersteuning; (testmanager)
- het rapporteren over de testvoortgang en kwaliteit van het testobject; (testmanager)
- gebruikers ondersteunen bij het bedenken van testgevallen (specifiek voor iteratief ontwikkelen). (tester)

3. Beschrijven organisatie

De samenhang tussen de genoemde rollen en de relaties met de andere betrokken partijen binnen het systeem ontwikkel proces moet bepaald en vastgelegd worden. De organisatie van de testsoort maakt vanzelfsprekend onderdeel uit van het grotere (project)geheel. In het geval van een project moet de testmanager ook de relatie naar een eventuele test- of kwaliteitsafdeling leggen.

Voor de organisatie van een testsoort zijn in grote lijnen de volgende mogelijkheden te onderkennen:

1. testen als **zelfstandige activiteit** of **geïntegreerd met andere activiteiten**;
2. testen in een **project-** of in een **lijnorganisatie** belegd; Deze keuzes zijn afhankelijk van de testsoort, project en organisatie. Soms, maar lang niet altijd, kan de testmanager hier invloed op uitoefenen. Voor een verdere toelichting van testen in de lijnorganisatie wordt verwezen naar paragraaf 8.3 “Permanente testorganisatie”.

	zelfstandige activiteit	**geïntegreerd**
projectorganisatie	acceptatietest, traditionele systeemtest	ontwikkeltests, systeemtest in agile omgeving
lijnorganisatie	testfabriek	onderhoudsproces

Figuur 6.7: Organisatie-indelingen met voorbeelden.

Onderstaand zijn de belangrijkste organisatievormen met enkele voordelen kort benoemd. De beschrijvingen en voordelen zijn nadrukkelijk als globale indicatie bedoeld, de praktijk laat vaak uitzonderingen zien.

Uitgediept

Testen als zelfstandige activiteit in project

Binnen het project is een team verantwoordelijk voor het inrichten en uitvoeren van de test. De testers binnen het team hebben als regel veel testkennis met daarnaast, afhankelijk van de testsoort, een mix van systeem- en

organisatiekennis.
Voordelen:

- Goede toegankelijkheid tot kennis van het systeem
- Goede afstemming tussen gebruikers, ontwikkelaars en testers
- Kennis en vaardigheden testers goed zichtbaar
- Focus op een doel en daardoor beter beheersbaar
- Onafhankelijk oordeel over de kwaliteit van het testobject

Testen geïntegreerd in project

Binnen het project werken testers, gebruikers en ontwikkelaars in hetzelfde team.

Vaak zijn meerdere teams actief. De tester is binnen zijn team verantwoordelijk voor het inrichten en uitvoeren van de test. De tester heeft als regel veel technische kennis van systeem en architectuur.

Voordelen:

- Uitstekende kennis van de applicatie en architectuur
- Nauwe samenwerking tussen gebruikers, ontwikkelaars en testers
- Zeer korte communicatielijnen
- Focus op een doel en daardoor beter beheersbaar

Testen als zelfstandige lijnorganisatie

Een aparte afdeling of organisatie heeft testen, zowel het inrichten als uitvoeren, als primaire taak. Projecten of andere lijnafdelingen besteden een bepaalde testopdracht uit aan deze afdeling/organisatie. Testkennis is dominant.

Voordelen:

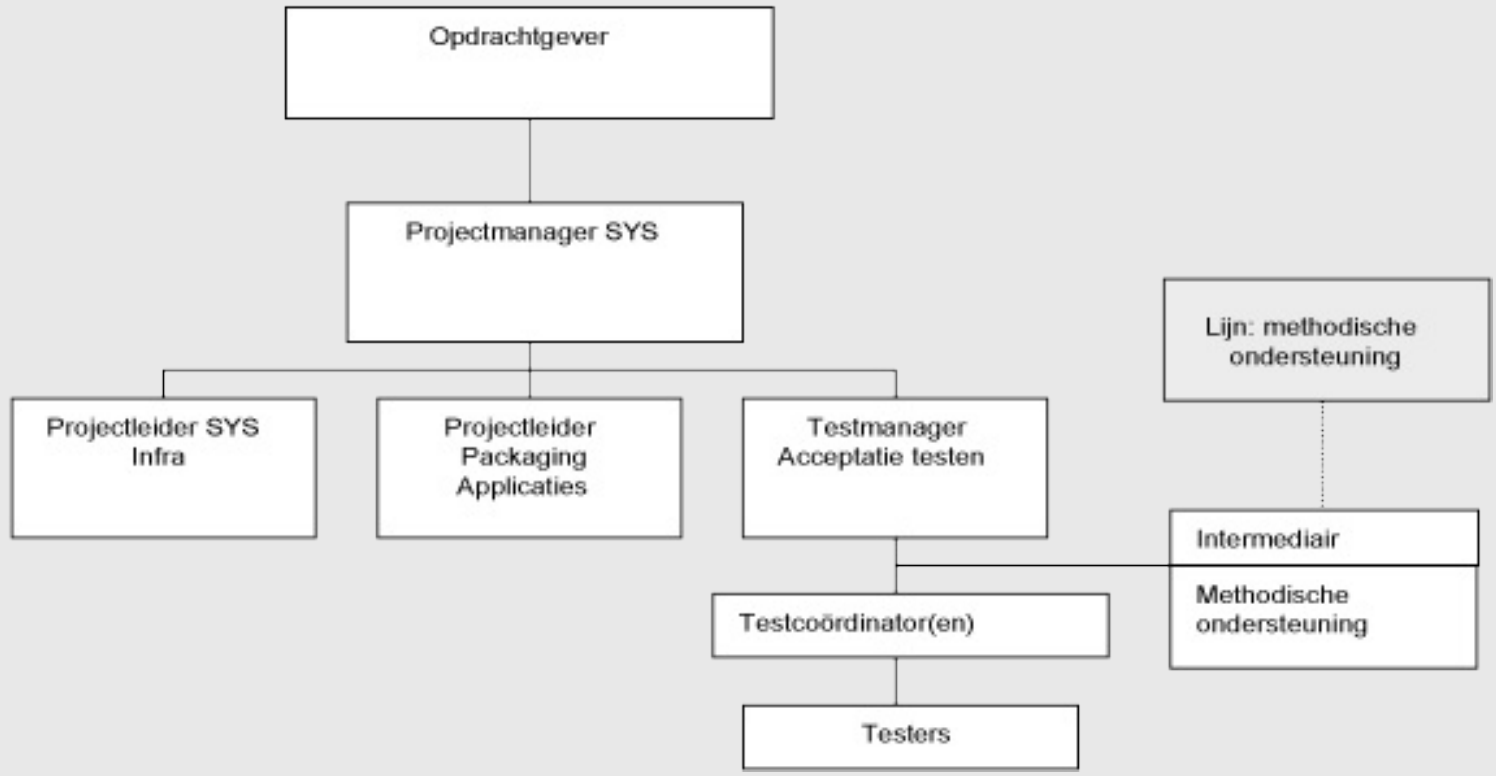
- Kennis en vaardigheden testers goed zichtbaar
- Onafhankelijk oordeel over de kwaliteit van het testobject
- Efficiëntiewinst door hergebruik en testautomatisering
- Permanent ingerichte infrastructuur maakt snelle start mogelijk
- Standaard ingericht testproces maakt snelle start mogelijk
- Verhoogde motivatie door carrière mogelijkheden testers

Testen geïntegreerd in lijnorganisatie

Binnen een ontwikkel- of beheerafdeling is de rol van tester vaak gecombineerd met andere rollen. De tester in deze organisatievorm heeft vaak veel kennis van systeem- en/of organisatie.

- Uitstekende kennis van het systeem en de organisatie
- Nauwe samenwerking tussen gebruikers, ontwikkelaars en testers
- Korte communicatielijnen
- Kennisbeheer over een applicatie is beter te realiseren

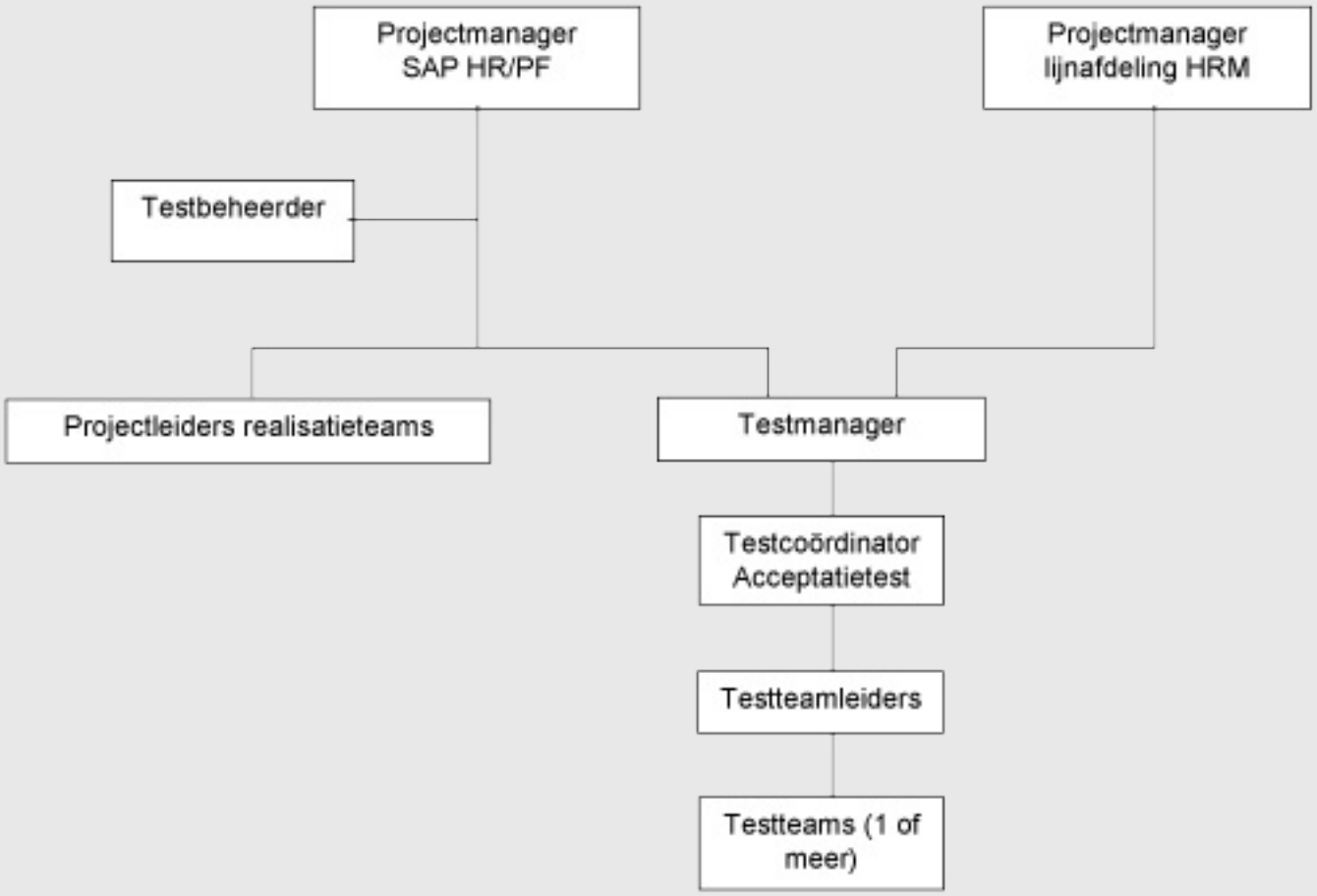
Onderstaand zijn enkele voorbeelden van organisatievormen uitgewerkt.



Figuur 6.8: Voorbeeld traditionele organisatie.

Bovenstaand een vrij traditionele organisatie waarbij de acceptatietestmanager onder de projectmanager valt en de systeemtest onder de projectleider (Packaging Applicaties). Testondersteuning wordt geleverd vanuit de lijn.

In onderstaand voorbeeld valt de testmanager zowel onder de projectmanager van het SAP-systeem als onder de projectmanager die het moet implementeren in de afdeling. Hoewel het hebben van 2 opdrachtgevers een ongewenste situatie is, is het in dit praktijkvoorbeeld goed gegaan. De testmanager heeft bij aanvang bedongen dat als beide opdrachtgevers het oneens zijn, zij er samen moeten uitkomen zonder de testmanager hierin als partij te betrekken. Dit is ook niet gebeurd.

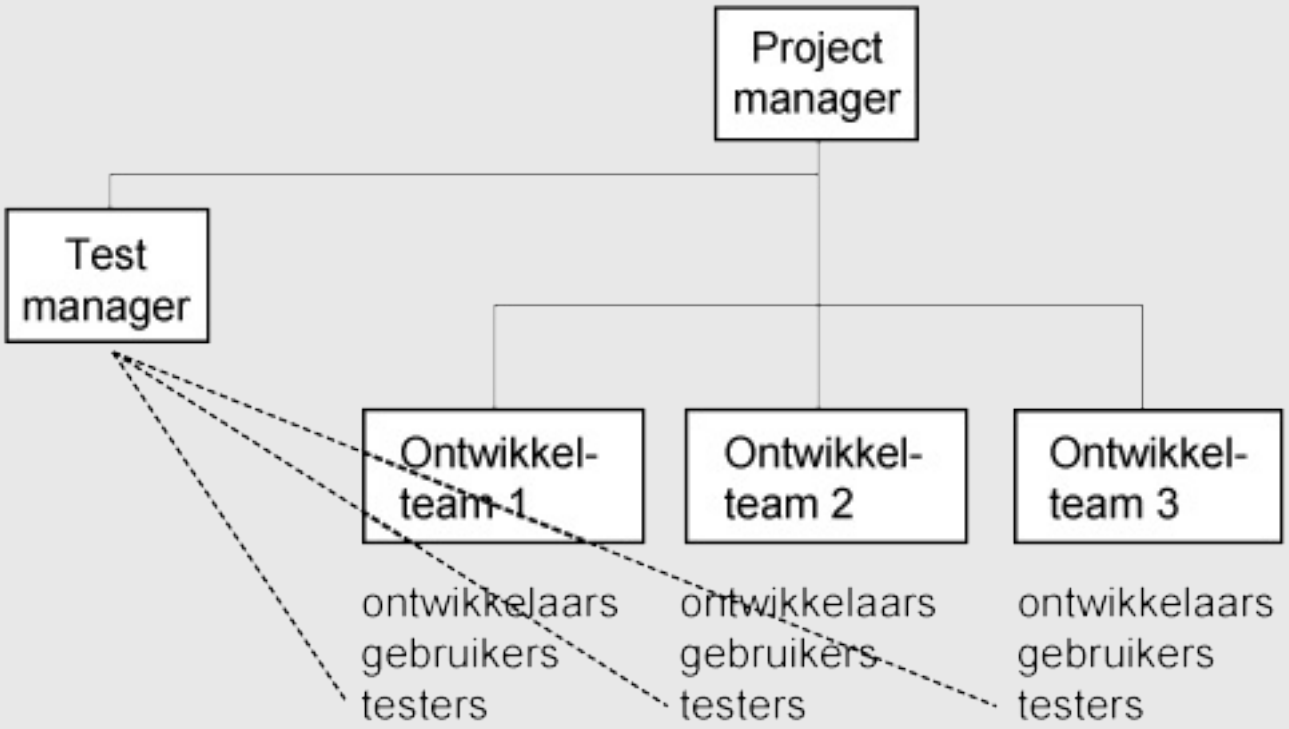


Figuur 6.9: Voorbeeld organisatie met twee managers.

Tip

Iteratieve en agile systeemontwikkeling

Als nadeel van geïntegreerd testen wordt genoemd dat dit een onafhankelijk kwaliteitsoordeel van de tester kan aantasten. Een mogelijke oplossing hiervoor is om de testmanager los van de ontwikkelteams te plaatsen, waarbij de testers in deze teams wél verantwoording dienen af te leggen aan deze. Het voordeel is dat de kloof tussen ontwikkelaar, gebruiker en tester in de teams zo klein mogelijk blijft, terwijl de testmanager toch kan signaleren indien het testen in een team in de knel komt door planningsdruk of andere omstandigheden. Dit vraagt van de testmanager inzicht in het totale product en project, gecombineerd met een goed politiek gevoel voor de balans 'kwaliteit' en 'tijdig deadlines halen'.



Figuur 6.10: Voorbeeld testmanager los van teams.

RACI

Zo nodig kan een RACI-tabel opgesteld worden met daarin activiteiten en betrokkenen tegen elkaar uitgezet. RACI staat voor Responsible, Accountable, Consulted en Informed. Op elk kruispunt kan aangegeven worden of een partij rechtstreeks verantwoordelijk (R) is, eindverantwoordelijk is (A), geraadpleegd (C) of geïnform moet worden, of helemaal niet.

Het is onmogelijk om één voorkeurorganisatie voor het testen te bepalen. In het algemeen geldt dat de structuur van de testorganisatie gelijk moet zijn aan die van het bijbehorende proces van systeemontwikkeling of pakketimplementatie. In veel gevallen betekent dit de projectorganisatie. Indien sprake is van herhaaldelijk testen in combinatie met schaarse (test)kennis, komt de in paragraaf 8.3 besproken permanente testorganisatie in aanmerking.

4. Toewijzen personeel

Nadat is vastgesteld welke testrollen er binnen het testproces vervuld moeten worden, kent de testmanager mensen toe aan elke rol. Hierbij houdt deze uiteraard rekening met hun beschikbaarheid en vaardigheden in relatie tot de voor de betreffende testrollen vereiste kennis en vaardigheden (zie paragraaf 8.6 "Testprofessionals" en hoofdstuk 16 "Testrollen"). Voor de duidelijkheid: de rollen hoeven niet per se te worden ingevuld met testprofessionals, ook eindgebruikers of ontwikkelaars kunnen bijvoorbeeld de rol van tester krijgen. Het gaat erom dat het team als geheel de juiste mix van kennis en vaardigheden krijgt op het gebied van het systeem, de organisatie en testen.

Overigens kan één persoon meerdere rollen vervullen, waarbij opgelet moet worden dat dit geen conflicterende verantwoordelijkheden geeft!

Tip

- Om grotere zekerheid te krijgen dat de testers voldoende testkennis hebben, kan specifiek gevraagd worden om gecertificeerde testers. Het EXIN organiseert een certificeringsprogramma specifiek voor TMap. Daarnaast organiseert de ISTQB (International Software Testing Qualifications Board) een internationaal en algemeen certificeringprogramma voor testen.
- Wanneer mensen van andere afdelingen of zelfs andere organisaties ingezet worden, moet de testmanager rekening houden met afspraken, procedures, selectieprocessen, enzovoort. Dit kan veel tijd kosten.
- Er is vaak externe pressie om bepaalde mensen als tester in het team te accepteren. Wanneer deze mensen niet geschikt zijn, moet de testmanager de rug recht houden en de consequenties in termen van hoge opleidings- en coachingkosten en lage productiviteit duidelijk maken.
- Het in dienst nemen of inhuren van een tester kan niet zomaar overgelaten worden aan een personeels- of inkoopafdeling. Goede informatie hierover is te vinden in [Rothman, 2006].

Naast de geschiktheid van de persoon voor de rol(len) is er nog een dimensie: die van het team. De natuurlijke neiging van de testmanager is om die personen te selecteren die hem qua persoonlijkheid het meest aanspreken. Dit kan leiden tot een team met op elkaar lijkende karakters. De theorie van teamvorming leert dat het juist de teams zijn met een mix van persoonlijkheden die de beste resultaten halen. Wellicht het bekendste model op dit gebied zijn de 9 teamrollen van Belbin (zie ook www. belbin.com). Daarbij wordt onderscheid gemaakt in functionele, organisatorische en persoonlijke rollen. Afhankelijk van de doelstelling heeft ieder team een ideale samenstelling.

Belbin onderscheidt de volgende rollen, met per rol een aantal kenmerken:

Plant	creatief, individualistisch, verbeeldingskracht, intellect, kennis
Voorzitter	kalm, zelfvertrouwen, nuchter, doelgericht, laat ieder teamlid tot zijn recht komen
Monitor/waarschuwer	strategisch inzicht, nuchter, weinig emoties, analyseert en bekritiseert
Bedrijfsman	plichtsgetrouw, behoudend, zet beslissingen om in werkzaamheden, praktisch, zelfdiscipline
Zorgdrager/afmaker	nauwgezet, zorgelijk, werkt achter de schermen
Onderzoeker	extravert, zoekt nieuwe mogelijkheden, enthousiast, communicatief
Vormer	dynamisch, energiek, extravert, ongeduldig
Groepswerker	sociaal gericht, coöperatief, kan goed luisteren, stimuleert en integreert
Specialist	vakman, solist, op vakinhoud gericht.

Voor verdere theorie hierover wordt verwezen naar [Belbin, 2003]. Een vertaling naar wat de beste testteamsamenstelling is, heeft [Roden, 2005] gegeven.

5. Vaststellen opleidingen en coachingsbehoefte

De mensen die betrokken worden bij de testsoorten moeten verschillende soorten kennis hebben, namelijk op het gebied van testen, de materie en het systeem.

- Voor testen kan gedacht worden aan: (de voordelen van) de testaanpak, strategiebepaling, de te hanteren testtechnieken en -tools;
- Bij materiekennis moet men bijvoorbeeld denken aan de organisatie en haar bedrijfsprocessen;
- Systeemkennis kan bestaan uit kennis over het ontwikkel- of implementatieproces, ontwerptechnieken, technische architectuur, (database of programmeer)tools, enzovoort.

Uitgediept

Kennis bijbrengen

Overigens is het niet de bedoeling enkel zeer ervaren testers in de teams te hebben die uitgebreide kennis op alle 3 vlakken hebben. Afhankelijk van de testsoort en de teamsamenstelling heeft elke persoon een bepaalde mix van deze soorten kennis nodig. Is de kennis van mensen op één van deze gebieden ontoereikend, dan moet deze kennis op peil gebracht worden. Een opleiding is hiervoor het meest voor de hand liggend. Hiervoor dient tijd en budget gereserveerd te worden. Timing is hierbij van belang: een opleiding is het meest effectief wanneer de opgedane kennis vrij snel daarna in de praktijk gebracht kan worden. Een opsomming van mogelijke soorten opleidingen is te vinden in paragraaf 8.6.6 “Opleidingen”. Na de eventuele opleiding is het zaak om mensen met onvoldoende kennis in het begin te laten coachen door iemand met ervaring. Dit versnelt het leerproces aanzienlijk. Vaak gebeurt dit

"on-thefly" in het testtraject, maar wanneer wordt ingeschat dat dit een substantiële activiteit is, moet dit ingepland worden en moeten er uren voor beschikbaar gesteld worden.

6. Vaststellen overleg- en rapportagestructuren

Vanuit het testproces moet met verschillende partijen gecommuniceerd worden. Voorbeelden van de partijen waarmee de testmanager communiceert, zijn:

- opdrachtgever
- testmanager van overkoepelende testproces
- projectmanagement (inclusief Change Control Board)
- acceptanten (gebruikersorganisatie, systeembeheer, functioneel beheer)
- stuurgroep
- projectleiders (ontwerp, bouw en/of implementatie)
- ontwikkelaars
- lijnorganisatie Testen
- kwaliteitsmanagement, QA
- accountancy, EDP-auditing.

Met elke partij moet afgesproken worden of overleg en/of rapportage plaatsvindt, en wat doel en frequentie ervan zijn.

Overlegvormen

Voor overlegvormen wordt afgesproken wie aanwezig zijn en wat eventueel de standaard agenda is.

Voorbeelden van overlegvormen voor de testmanager zijn:

- wekelijks overleg met alle andere testmanagers onder leiding van de testmanager van het overkoepelende proces;
- wekelijks projectoverleg;
- wekelijks overleg Change Control Board; bevindingenoverleg (standaard 1 x per week, tijdens testuitvoering 3 x per week)
- wekelijks testteamoverleg;
- dagelijkse stand-up meeting.

Voorbeeld van een vaste agenda voor een testteamoverleg:

Agendapunt	Onderwerp	Tijd	Wie
1.	Opening - Vaststellen agenda - Mededelingen	xx.xx – xx.xx	<testmanager>
2.	Notulen vergadering d.d.: <xx-xx-xxxx>	xx.xx – xx.xx	Allen
3.	Actielijst d.d.: <xx-xx-xxxx>	xx.xx – xx.xx	Allen
4.	Status, voortgang en kwaliteit: - testeenheid 1 - testeenheid 2 - testeenheid 3 - …	xx.xx – xx.xx	<Tester 1> <Tester 2> <Tester 3> …
5.	Kwaliteit testproces - <Wat gaat goed en wat kan beter?> <TPI status> <Bevindingenbeheer> <Testwarebeheer> <Reviews> - …	xx.xx – xx.xx	Allen
6.	Rondvraag	xx.xx – xx.xx	Allen

Rapportages

Vanuit de BDTM-visie vindt rapportage plaats op de vier aspecten Resultaat, Risico's, Tijd en Kosten.

- Resultaat
  - De uitkomst van uitgevoerde tests op het niveau van kenmerk/deelobject;
  - Het resultaat in termen van wel/niet gehaalde testdoelen (bedrijfsprocessen, user reqs, …);

  - Eventuele trendanalyses;
- Risico
  - Signalering van onderdelen die lichter (of niet) worden getest dan de risico-inschatting aangeeft en daarmee een hoger risico geven;
  - Gesignaleerde (test)projectrisico's;
- Tijd + Kosten
  - Voortgang testen (in activiteiten, producten, bestede uren en optioneel geld, datums);
  - Voorspelling wanneer testen klaar is.

Rapporteren over risico's en resultaat vindt plaats op het met de opdrachtgever en andere belanghebbenden afgesproken niveau van testdoelen. De risicotabellen van de productrisicoanalyse zijn met dit doel beheerd. Het is aan de testmanager om testresultaten op kenmerken/deelobjecten op een goede manier en aan de hand van de tabellen te vertalen naar dit niveau.

Rapportage kan op diverse manieren plaatsvinden, naar verschillende doelgroepen en op verschillende momenten. De belangrijkste vormen van rapportage zijn:

- Voortgangs- en kwaliteitsrapportage
  Informatie en advies over voortgang (en optioneel kwaliteit van) testproces en kwaliteit/risico's van het testobject, gebaseerd op de vier BDTM-aspecten.
  Frequentie: periodiek, bij voorkeur wekelijks.
- Risicorapportage
  Bij bepaalde (project)risico's kan, hetzij op verzoek, hetzij op eigen initiatief, de testmanager rapporteren over het risico, de gevolgen voor het testproces en mogelijke maatregelen om met het risico om te gaan. In projectmanagementmethode Prince2 worden dit ook wel exception reports genoemd.
  Frequentie: ad-hoc.
- Vrijgaveadvies
  Informatie en advies over kwaliteit/risico's van het testobject + formeel vastgelegd vrijgaveadvies.
  Frequentie: tegen het einde van de testuitvoering, vóór de beslissing tot vrijgave moet worden genomen.
- Eindrapportage
  Evaluatie testproces en testobject, terugkijkend vanaf originele plan.
  Frequentie: eenmalig, aan het eind van het testproces

Van elk van deze rapportagevormen bepaalt de testmanager naar wie deze verstuurd moet worden, ter goedkeuring of ter informatie, met welke inhoud en mate van detail, en met welke frequentie. In de activiteit "Oriënteren opdracht" heeft de testmanager al gekeken welke partijen rapportages moeten/willen ontvangen. In overleg met de opdrachtgever wordt dat nu nader ingevuld. Als hulpmiddel om snel te overzien wie welk rapport moet krijgen kan een matrix opgesteld worden van rapportagevormen en doelgroepen. De rapportagevormen en -inhoud worden gedetailleerd besproken in paragraaf 6.3.3 "Rapporteren".

Uitgediept

Inhoudelijk is het voortgangs- en kwaliteitsrapport het meest van belang, omdat op basis van deze informatie en adviezen nog tijdig (bij)gestuurd kan worden. De gegevens hiervoor worden aangeleverd door beheer in te richten, zie ook paragraaf 6.2.12 "Inrichten beheer". De rapportage dient gegevens te bevatten over de meest recente rapportageperiode en gecumuleerde gegevens over het gehele testproces.

Producten

Een beschrijving van de testorganisatie, vastgelegd in het testplan.

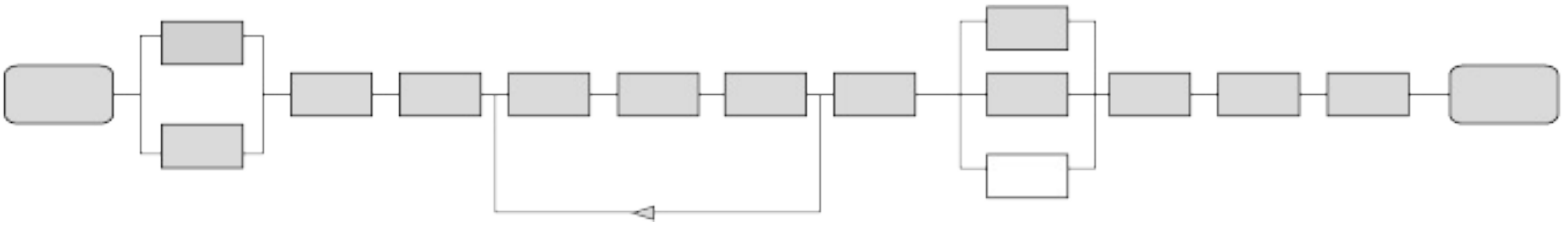
Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.2.11 Definiëren infrastructuur



Doel

Het vaststellen van de voor het testproces benodigde infrastructuur.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Definiëren testomgeving;
2. Definiëren testtools;
3. Definiëren kantoorinrichting;
4. Vaststellen infrastructuurplanning.

Uitgebreide beschrijvingen van testomgevingen en testtools zijn te vinden in respectievelijk paragraaf 8.4 en 8.5.

1. Definiëren testomgeving

Elke testsoort heeft een testomgeving nodig om de tests uit te voeren. Deze omgeving is op hoofdlijnen samengesteld uit de volgende componenten:

- hardware,
- software,
- koppelingen,
- omgevingsdata,
- beheertools en
- processen.

De omgeving moet dusdanig worden samengesteld en ingericht dat aan de hand van de testresultaten optimaal kan worden vastgesteld in hoeverre het testobject aan de gestelde eisen voldoet. De omgeving heeft grote invloed op de kwaliteit, doorlooptijd en kosten van het testproces.

Om de testomgeving goed te kunnen beheersen, is deze dan ook vaak apart van de ontwikkel- of productieomgeving. Hierbij stelt elke testsoort andere eisen aan haar omgeving.

Op www.tmap.net is een checklist “Testomgevingen” te vinden die behulpzaam kan zijn bij het definiëren van de testomgeving.

Wanneer de testomgeving al bestaat, bijvoorbeeld in een onderhoudstraject, kan volstaan worden met hiernaar te verwijzen en de eventuele te maken aanpassingen te benoemen.

2. Definiëren testtools

De benodigde testtools worden vastgesteld. Testtools kunnen ondersteuning bieden bij de meeste testactiviteiten (zie ook paragraaf 8.5 “Testtools”).

Naast de (over)bekende testtools zoals testwarebeheer-, record&playback- en bevindingenbeheertools, moet ook gedacht

worden aan kleine, freeware of zelf te bouwen tooltjes. Dergelijke tools kunnen vaak tegen een kleine investering in tijd geïmplementeerd worden en daarna veel ondersteuning geven. Om freeware-tools op te sporen is het internet onontbeerlijk (zoek bijvoorbeeld op “freeware test tool”) . Voor zelf te bouwen tools is het verstandig met de ontwikkelaars te overleggen: vaak hebben die al dergelijke tooltjes, anders kunnen ze mogelijk tegen een geringe inspanning worden gemaakt.

Uitgediept

Aangezien tools ondersteuning moeten bieden aan het testproces, lijkt de logische volgorde om eerst het proces te definiëren en dan het tool te selecteren, “structure before tool”. Dit is echter niet helemaal waar. Sommige heel nuttige tools (met name voor testwarebeheer en record&playback) stellen eisen aan het proces, bijvoorbeeld de wijze waarop testgevallen worden vastgelegd. Wanneer de testmanager hier geen rekening mee houdt, is het tool niet (efficiënt) inzetbaar. Beter is het dus om procesinrichting en toolselectie enigszins gelijk op te laten gaan.

3. Definiëren kantoorinrichting

De voor testen benodigde kantoorinfrastructuur (werkkamers, vergaderkamers, telefoons, PC’s, netwerkaansluitingen, kantoorsoftware, printers, enzovoort) wordt in grote lijnen gedefinieerd. Het gaat hierbij om kantoorinrichting in de breedste zin, want ook de testers moeten hun werk onder goede omstandigheden kunnen uitvoeren. Een checklist voor de kantoorinrichting is te vinden op www.tmap.net.

Het goed en tijdig regelen van de kantoorinfrastructuur heeft tot gevolg dat allerlei efficiëntieverliezen, zoals verhuizingen, wachttijden en improductieve uren, tot een minimum beperkt blijven. Een slecht voorbeeld in dit verband is wanneer de testers fysiek op te grote afstand van elkaar en de rest van het project moeten zitten. De inrichting van de werkplekken heeft daarnaast ook positieve invloed op de kwaliteit van het testproces. Denk in dit verband aan de kwaliteit van zowel interne als externe communicatie en aan de motivatie en productiviteit van de mensen.

Tips

- Informeer in een zo vroeg mogelijk stadium naar de besteltijden van de diverse benodigdheden.
- Zorg dat eventuele verhuizingen en dergelijke apart worden begroot.
- Als testers op fysiek grotere afstand van elkaar zitten, kunnen mogelijk extra uren voor overhead begroot worden. Dit maakt de nadelen van de gekozen kantoorinfrastructuur duidelijker.

4. Vaststellen infrastructuurplanning

De testmanager legt de gemaakte afspraken vast en stelt een globale planning op met daarin de tijdstippen van het ter beschikking stellen van de diverse faciliteiten. Het verder bestellen en regelen van de infrastructuur valt onder verantwoordelijkheid van de testinfrastructuurcoördinator.

Producten

De beschrijving van de benodigde infrastructuur, inclusief planning, vastgelegd in het testplan.

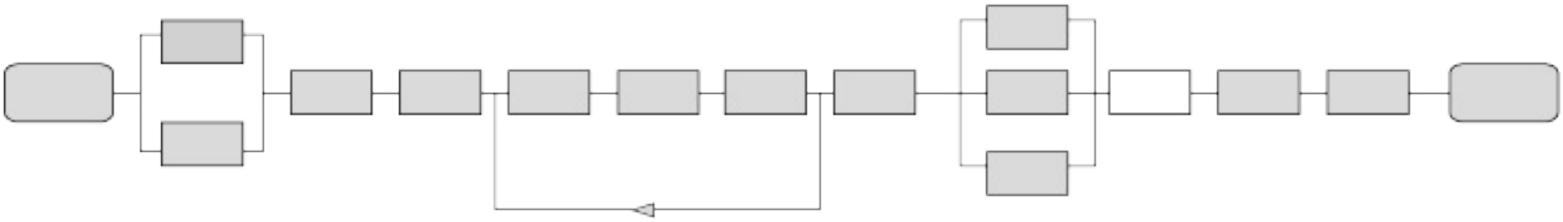
Technieken

Checklist “Testomgevingen” (www.tmap.net).
Checklist “Kantoorinrichting” (www.tmap.net).

Tools

Niet van toepassing.

6.2.12 Inrichten beheer



Doel

Het vastleggen van de wijze waarop het beheer van het testproces, de infrastructuur, de testproducten en de bevindingen wordt geregeld.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Definiëren testprocesbeheer;
2. Definiëren infrastructuurbeheer;
3. Definiëren testproductbeheer;
4. Definiëren bevindingenprocedure.

Op testplan niveau kunnen hiervoor normen en standaards worden opgesteld, ondersteund door procedures, sjablonen en tools (testwarebeheertools, plannings- en voortgangsbewakingstools). Soms zijn er op overkoepelend voorzieningen getroffen waar gebruik van gemaakt kan/moet worden.

1. Definiëren testprocesbeheer

Testprocesbeheer richt zich op het beheren van het testproces in termen van voortgang en kwaliteit, en het inzicht geven in de kwaliteit van het testobject. Hiertoe moet identificatie, registratie, administratie, opslag en interpretatie van de volgende gegevens plaatsvinden:

- voortgang en de besteding van budget en tijd;
- kwaliteitsindicatoren;
- teststatistieken.

Het beheer wordt soms belegd bij een aparte rol: testprojectadministrateur (zie paragraaf 16.3 “Rollen niet als functie beschreven”).

Deze informatie vormt de basis voor sturing en rapportage door testmanagement. Omdat beheersing van het testproces steeds belangrijker wordt, worden hoge eisen aan het beheer van met name het testproces gesteld. Er moet snel, liefst real-time, inzicht zijn in de actuele stand van zaken. In dit verband wordt ook wel de term “**dashboard**” gebruikt: een simpel overzicht waaruit alle overbodige informatie is weggehaald en dat in één oogopslag de belangrijkste informatie geeft: kwaliteit van het testobject (in termen van bevindingen) en voortgang van het testproces. Plannings- en voortgangsbewakingstools maar ook testwarebeheertools kunnen hier uitstekend bij ondersteunen.

Onderstaand een voorbeeld:

	Regressie	Deelsys1	Deelsys2	Deelsys3
User stories (test basis)				
Totaal	nvt	103	23	nvt
Status 1 (bedacht)	nvt	62	23	nvt
Status >= 2 (opgepakt door testen)	nvt	41	0	nvt
Handmatige testscripts				
Totaal gepland	291	145	35	111
Gereed	291	62	0	93
Nog aan te passen	0	63	14	18
Nog te maken	0	20	21	0

Resultaten laatste handmatige testronde				
Datum	nvt	nvt	nvt	nvt
Totaal gelopen	nvt	nvt	nvt	nvt
OK	nvt	nvt	nvt	nvt
Niet OK	nvt	nvt	nvt	nvt
Niet gelopen	nvt	nvt	nvt	nvt
Niet voltooid	nvt	nvt	nvt	nvt
Geautomatiseerde testscripts				
Totaal gepland	240	nvt	nvt	111
Gereed	180	nvt	nvt	1
Nog aan te passen	180	nvt	nvt	0
Nog te maken	60	nvt	nvt	110
Resultaten laatste geautomatiseerde testronde				
Datum	18-12-05	nvt	nvt	17-12-05
Totaal gelopen	111	nvt	nvt	1
OK	43	nvt	nvt	1
Niet OK	66	nvt	nvt	0
Niet gelopen	2	nvt	nvt	0
Niet voltooid	0	nvt	nvt	0
Bevindingen				
Nieuw	9	13	nvt	0
Open	197	1	nvt	22
In oplossing	20	5	nvt	0
Te hertesten	14	0	nvt	1
Afgesloten	274	5	nvt	143

(nvt = niet van toepassing)

Voortgang en besteding van budget en tijd

De voortgangsinformatie biedt de opdrachtgever en het testmanagement inzicht in het testproces op basis waarvan het testproces zo nodig bijgestuurd kan worden. Bij negatieve trends kunnen tijdig maatregelen genomen worden.

De te beheren onderdelen zijn de activiteiten en/of producten, gerelateerd aan uren, resources, doorlooptijd en met onderlinge afhankelijkheden.

Uitgediept

De meeste activiteiten resulteren in één of meer producten, zoals (master)testplan, rapportages, testscripts, testbestanden, testlogs, enzovoort. Uitzondering zijn ondersteunende activiteiten, die meestal geen tastbaar product opleveren.

Er moet een keuze worden gemaakt om op het niveau van activiteiten of op het niveau van producten de voortgang te registreren, waarbij ook de mix-vorm mogelijk is.

Het voordeel van beheer op producten is dat dit beter meetbaar is dan activiteiten: het is makkelijker te beoordelen of een product voor 80% klaar is dan een activiteit. Ook steeds meer ontwikkel- en projectmanagementmethodes sturen op producten.

Bij de identificatie van activiteiten of producten moet gelet worden op de gewenste mate van detail. Is het belangrijk om een activiteit van enkele uren apart te registreren of is het efficiënter deze te registreren als onderdeel van een grotere activiteit? Dit wordt in overleg met de opdrachtgever bepaald.

Kwaliteitsindicatoren

Het doel van het testen is om informatie en advies te geven over de risico's en kwaliteit van het te testen object. Om deze informatie te kunnen geven, worden kwaliteitsindicatoren bijgehouden. De bekendste en meest voor de hand liggende indicator is de bevinding. Door allerlei gegevens bij een bevinding vast te leggen, zoals bijvoorbeeld status, ernst, oorzaak, kwaliteitsattribuut en systeemdeel, kan later allerlei kwalitatieve informatie uit de bevindingen gehaald worden. Denk hierbij aan het aantal openstaande bevindingen voor een bepaald deel van het systeem, het aantal gedane bevindingen in een bepaalde periode, het aantal bevindingen op de requirements, enzovoort. Voor meer informatie over bevindingen wordt verwezen naar hoofdstuk 12 "Bevindingenbeheer". Daarnaast zijn nog diverse andere indicatoren

mogelijk. Te denken valt aan het aantal hertests of het aantal storingen binnen de testinfrastructuur (als indicator voor de betrouwbaarheid ervan).

De hierboven genoemde indicatoren zeggen iets over de kwaliteit van het testobject. Een andere groep van indicatoren zegt iets over de kwaliteit van het testproces zelf. Hierbij moet men denken aan (zie ook paragraaf 6.3.3 “Rapporteren”):

effectiviteit van testen	worden de (belangrijke) fouten gevonden?
efficiëntie van testen	worden de fouten zo snel en goedkoop als mogelijk gevonden?
controleerbaarheid van testen	verloopt het testproces transparant en op de afgesproken wijze?

Teststatistieken

Op basis van bovenstaande gegevens bouwt de testmanager statistieken op.

Statistieken kunnen cijfermatig de voortgang van het testproces en kwaliteit van het testobject inzichtelijk maken, inclusief eventuele trends. Ook kunnen statistieken aangelegd worden over de kwaliteit van het testproces zelf.

Tips

- Het vaststellen welke gegevens kunnen worden gemeten (metrics) is uitgebreid beschreven in hoofdstuk 13 “Metrics”.
- Wanneer er een lijnorganisatie Testen bestaat, is het aan te bevelen hiermee te overleggen over bij te houden statistieken. Mogelijk kan de lijnorganisatie informatie geven welke statistieken belangrijk zijn in de organisatie, mogelijk kunnen ze ondersteuning bieden. Andersom is de lijnorganisatie als het goed is geïnteresseerd in de statistieken vanuit het project.

2. Definiëren infrastructuurbeheer

De testinfrastructuur is onderverdeeld in drie groepen faciliteiten:

- testomgeving;
- testtools;
- kantoorinrichting.

De testinfrastructuur wordt tijdens de vroege fasen van het testproces gespecificeerd en besteld. Na installatie, intake en acceptatie ervan dient de infrastructuur beheerd te worden. In de praktijk wordt het beheer meestal uitbesteed aan een afdeling als systeembeheer of operations, waarbij al of niet de testinfrastructuurcoördinator het communicatiekanaal vormt tussen testproces en beherende afdeling.

Uitgediept

Ten aanzien van de verdeling van de beheertaken zijn de diverse aspecten van de testinfrastructuur in twee groepen te verdelen:

Technisch beheer

- testomgevingen (hard- en software; beheerprocedures);
- testbestanden (fysiek);
- netwerken voor testomgeving en kantoorinrichting;
- technische kantoorinrichting;
- testtools;
  De belangrijkste taken zijn:
  – Versiebeheer
  – Configuratiebeheer
  – Oplossen knelpunten
  – Beschikbaar stellen logging
  – Back-up & restore
  – Recovery

– (Technisch) monitoren
– Autorisaties verlenen
– Beschikbaarstellen
– Doorvoeren wijzigingen
– Onderhoud
– Storingsbehandeling.

De technische beheertaken die moeten worden uitgevoerd, zijn onderdeel van de rol testinfrastructuurcoördinator. Bij de uitvoering van deze taken wordt zonodig ondersteuning verleend door de leverancier of een afdeling als systeembeheer of infrastructuur services

Logistiek beheer

- het niet technische deel van de kantoorinrichting, zoals kantinevoorzieningen, vervoer, toegangspasjes, enzovoort.

De taken in het kader van het logistiek beheer zijn niet testspecifiek en worden daarom in dit boek niet verder behandeld.

3. Definiëren testproductbeheer

Op testplan niveau worden voor het beheer van de testproducten normen en standaards opgesteld, ondersteund door procedures, sjablonen en tools. Dit bevordert de herbruikbaarheid van de producten en de communicatie erover. Het is raadzaam aan te sluiten op de algemeen binnen het systeemontwikkel proces geldende normen en standaards voor documentatie en configuratiebeheer.

Het beheer wordt soms belegd bij een aparte rol: testwarebeheerder (zie paragraaf 16.3 “Rollen niet als functie beschreven”).

De volgende te beheren groepen van producten kunnen onderscheiden worden:

- Producten zoals testware en testprojectdocumenten. Aan herbruikbare producten zoals testware worden doorgaans hogere eisen aan het beheer gesteld, bijvoorbeeld dat versies bewaard blijven. Zie ook paragraaf 6.2.9 “Definiëren testproducten”.
- Externe producten zoals de testbasis en het te testen object. De verantwoordelijkheid voor het beheer hiervan ligt buiten het testproces. Wel is het belang van goed (versie)beheer voor het testproces heel groot. Om die reden worden vaak eisen vanuit het testplan aan het externe beheer gesteld, bijvoorbeeld dat elk product uniek identificeerbaar is.

Er moet een keus gemaakt worden welke producten beheerd gaan worden en in welke mate. Het beheer kan goed ondersteund worden door middel van een testwarebeheertool.

Uitgediept

Onderstaand is een aanzet tot een procedure Testproductbeheer. De procedure bestaat uit vier stappen:

Aanleveren

De te beheren producten worden door de testers bij de beheerder aangeleverd. Bij voorkeur worden de aangeleverde bestanden in een aparte directory geplaatst. De producten moeten compleet worden aangeleverd (onder andere voorzien van versiedatum en versienummer). De beheerder controleert op volledigheid. Hieronder volgen enkele zaken waarop gecontroleerd kan worden:

- Naam auteur;
- Soort document (ook in naam document);
- De definitieve versie en versiedatum;
- Juistheid van verwijzingen naar andere documentatie (de testproducten moeten duidelijk verwijzen naar het bijbehorende testobject en de bijbehorende testbasis);
- Mutatieoverzicht: Overzicht van de versies, versiedata en reden van wijziging inclusief naam van de persoon die de wijzigingen heeft aangebracht.

- Producten in elektronische vorm moeten worden aangeleverd met een vaste naamgeving, waarin tevens het versienummer verwerkt is.

Registreren

De beheerder registreert de aangeleverde producten in zijn administratie aan de hand van onder andere naam leverancier, naam product, datum en versienummer. Tevens wordt opgenomen hoe lang de betreffende producten bewaard dienen te worden. In bepaalde gevallen kan het nodig zijn om ook de informatie van de gerelateerde producten op te nemen voor het te registreren product. Dit treft men aan bij organisaties waar traceerbaarheid een belangrijk issue is, bijvoorbeeld vanwege wettelijke verplichtingen.

Bij het registreren van gewijzigde producten moet de beheerder erop toezien dat de consistentie tussen de verschillende producten gewaarborgd blijft.

Archiveren

Er wordt onderscheid gemaakt tussen nieuwe en gewijzigde producten. Grofweg gezegd worden nieuwe producten toegevoegd aan het archief en vervangen gewijzigde producten de vorige versie.

Raadplegen

Uitgifte van producten aan projectteamleden of derden geschiedt door middel van een kopie van de gevraagde producten. De beheerder registreert welke versie van de producten aan wie is uitgereikt en wanneer.

Uitgediept

Traceerbaarheid

Het wordt, mede als gevolg van regel- en wetgeving (IFRS, Code Tabaksblat, SOX, FDA (Food and Drug Administration) en FAA (Federal Aviation Administration)), steeds belangrijker om aan te geven *dat* er getest is en *wat* er getest is. Het aantonen *wat* er getest is, wordt bereikt met traceerbaarheid (aantonen welke testgevallen op welk gedeelte van de testbasis betrekking hebben). Het bewijs *dat* er getest is, moet geleverd worden door expliciete verslaglegging. Vervolgens wordt geëist dat de bevindingen aantoonbaar (met bewijs) worden afgehandeld. Wil men aan deze strenge eisen van traceerbaarheid en bewijslast voldoen, dan dienen het testproductbeheer, het bevindin gen beheer en de kwaliteitszorg voor het testen hierop toegesneden te zijn (en dient er extra budget voor vrijgemaakt te worden!). Het testbeheer dient dan zodanig van opzet te zijn dat per stap de traceerbaarheid en bewijslast te volgen is.

Dat betekent dat:

- In de testspecificaties duidelijk aangegeven is van welk deel van de testbasis deze is afgeleid;
- Bij de testuitvoering de bewijslast geleverd wordt welke testgevallen daadwerkelijk uitgevoerd zijn;
- Inzichtelijk wordt gemaakt welke testgevallen tot welke bevindingen hebben geleid;
- De bewijslast tijdens de hertest wordt vastgelegd; welke bevindingen zijn opgelost en in hertest goed bevonden.

Daarnaast heeft traceerbaarheid nog de volgende grote voordelen voor testen:

- Er is veel inzicht in de kwaliteit en diepgang van de test, omdat van de requirements, het functioneel en het technisch ontwerp en de programmatuur bekend is met welke testgevallen deze gecontroleerd zijn (of worden). De kans op omissies in de test wordt hierdoor veel kleiner.
- Bij veranderingen in de testbasis of het testobject is heel snel te herleiden welke testgevallen aangepast en/of opnieuw uitgevoerd moeten worden.
- Als het onder hoge tijdsdruk niet mogelijk is om alle geplande tests uit te voeren, zullen testgevallen geschrapt moeten worden. Omdat de relatie met requirements, specificaties en software bekend is, kunnen die testgevallen vervallen waarvan de bijbehorende requirement of specificatie het minste risico oplevert in productie en is duidelijk bij welke requirements of specificaties er geen of een minder gefundeerde uitspraak over de kwaliteit mogelijk is.

Wil men de test traceerbaar inrichten, dan is de inzet van tools voor testproductbeheer vrijwel onmisbaar.

4. Definiëren bevindingenprocedure

Er dient een bevindingenprocedure te worden opgesteld, waarmee bevindingen kunnen worden afgehandeld en beheerst. Met grote voorkeur wordt deze procedure ondersteund door een tool. Omdat een bevindingenprocedure voor het gehele project geldt en niet voor een afzonderlijke testsoort, kan deze procedure het beste op mastertestplan-niveau worden gedefinieerd. Dit maakt het ook mogelijk om testsoort-overstijgende trends te signaleren. Een beschrijving van de bevindingenprocedure is opgenomen in hoofdstuk 12 “Bevindingenbeheer”. Het beheer wordt soms belegd bij een aparte rol: bevindingenbeheerder (zie paragraaf 16.3 “Rollen niet als functie beschreven”).

Producten

Een beschrijving van de diverse beheerprocessen, vastgelegd in het testplan.

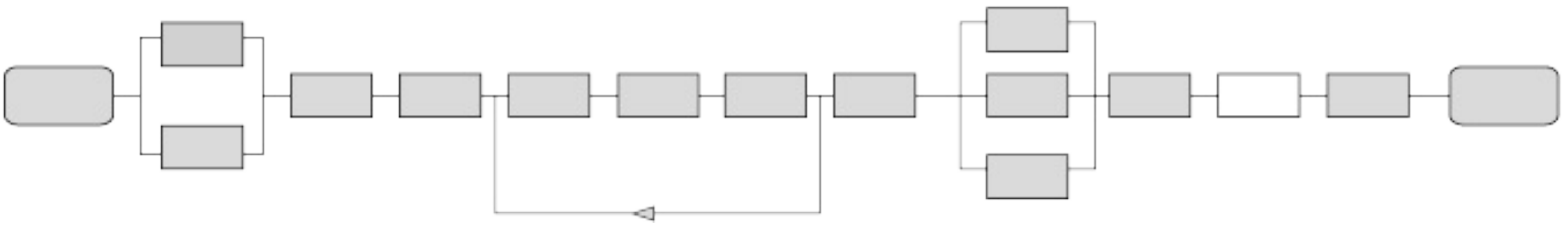
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

6.2.13 Bepalen testprocesrisico’s (& maatregelen)



Doel

Het expliciet benoemen van de risico’s voor de testsoort. Hierdoor krijgen opdrachtgever en andere betrokkenen een beter begrip van de risico’s voor de test en kunnen daar in de sturing van het totale voortbrengingsproces rekening mee houden.

Werkwijze

Bij het uitvoeren van voorgaande activiteiten heeft de testmanager een beeld gekregen van de (on)mogelijkheden voor het testproces, maar ook van bedreigingen en risico’s. In het testplan wordt per risico aangegeven of en zo ja, welke maatregelen zijn genomen ter afdekking of vermindering van het gesignaleerde risico. Denk hierbij aan preventieve maatregelen om risico’s te vermijden, maar eventueel ook aan detectieve maatregelen om problemen tijdig te kunnen signaleren. Vervolgens worden deze risico’s bewaakt tijdens het beheer van het testproces.

Beseft moet worden dat deze stap niet meer is dan het bedenken van de risico’s zoals die *aan het begin* van het traject bekend zijn. Daarna neemt de testmanager deze risico’s op in de voortgangsrapportage in de separate paragraaf “Projectrisico’s”. Vervolgens worden deze risico’s gevolgd, bewaakt, afgevoerd, nieuwe risico’s gesignaleerd, enzovoort. Wanneer deze activiteit ook op projectniveau plaatsvindt, kan het hiermee gecombineerd worden.

Uitgediept

De risico's kunnen onder meer betrekking hebben op:

- *Realiteitswaarde planningen*
  De testplanning is afhankelijk van de planningen van de diverse andere partijen. Hoe reëel zijn deze planningen?
- *Ingangskwaliteit*
  De twee belangrijkste vormen van input voor het testproces zijn de testbasis en het testobject. Als deze input van onvoldoende kwaliteit is, werkt dit zeer verstorend voor het testproces.
- *Resources*
  Voor het testen zijn mensen en middelen nodig, in een bepaalde hoeveelheid en van een bepaalde kwaliteit. In de praktijk blijkt vaak dat de in het plan afgesproken resources bij uitvoering niet (volledig) of op tijd geleverd kunnen worden.
- *Stabiliteit*
  In welke mate gaat de testbasis gedurende het testproces nog veranderen? Des te meer wijzigingen, des te groter de gevolgen voor het testproces in termen van herstelwerk.
- *Infrastructuur*
  Is deze stabiel voor de test, moet de omgeving gedeeld worden met andere partijen, is de omgeving voldoende representatief, is voldoende ondersteuning beschikbaar? In veel testprojecten vormt de infrastructuur het meest onbeheersbare risico.

Producten

Een beschrijving van de gesignaleerde (test)projectrisico's en mogelijke maatregelen, vastgelegd in het testplan.

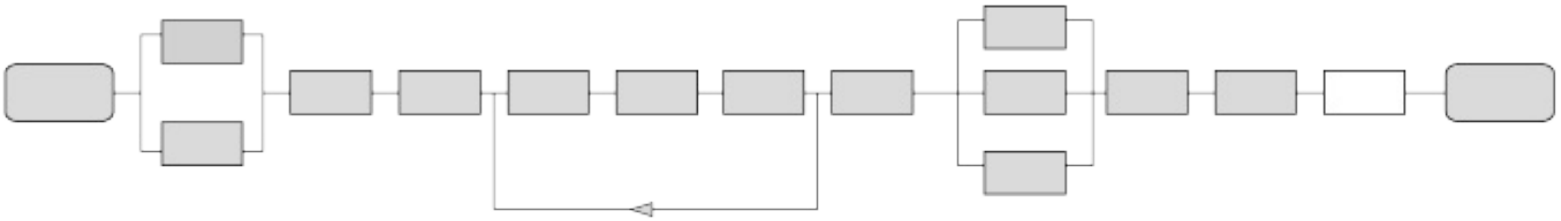
Technieken

Checklist "Risico's testproces" (www.tmap.net).

Tools

Niet van toepassing.

6.2.14 Terugkoppelen en fixeren plan



Doel

Enerzijds het vastleggen van de resultaten van alle tot nu toe uitgevoerde activiteiten en anderzijds het verkrijgen van goedkeuring van de gekozen aanpak door de opdrachtgever.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Opstellen testplan;
2. Terugkoppeling testplan;
3. Fixeren testplan.

1. Opstellen testplan

De resultaten van alle tot nu toe uitgevoerde activiteiten worden vastgelegd in het testplan. Het testplan bevat in ieder geval de volgende hoofdstukken:

- Opdrachtformulering;
- Teststrategie
- Aanpak (testeenheden, met te hanteren testvormen en -technieken per testeenheid);
- Organisatie;
- Infrastructuur;
- Beheer;
- Bedreigingen, (project)risico's en maatregelen;
- Begroting en planning.
- bijlage: Productrisicoanalyse

2. Terugkoppeling testplan

De verschillende onderdelen uit het plan moeten consistent zijn met elkaar.

In de praktijk verloopt het opstellen van een consistent plan in een aantal slagen. Het testplan met de resultaten van voorgaande activiteiten wordt teruggekoppeld met de opdrachtgever en andere betrokken partijen (zoals de testmanager van het overkoepelende testproces) voor goedkeuring of bijstelling. Dit maakt de te volgen testaanpak transparant en stuurbaar, geheel in lijn met BDTM.

Tip

Sommige testmanagers hebben er goede ervaring mee om het plan in een walkthroughsessie met de belangrijkste betrokkenen door te nemen. Eventuele tegenstellingen komen zo snel naar boven, zodat het aantal terugkoppelslagen tot een minimum beperkt kan worden.

Door de strategie aan te passen (waarbij de risicoanalyse in principe niet verandert), kan de testmanager de opdrachtgever laten sturen op de afweging tussen testinspanning versus testzwaarte. Dit resulteert in een aangepaste strategie, waarbij het schrappen of toevoegen van testzwaarte wordt getoond door respectievelijk ○ of ◒ in plaats van ●, zoals al eerder aangegeven bij paragraaf 6.2.7 "Bepalen planning". De testmanager moet de gevolgen voor begroting, planning én risico's van deze aanpassing duidelijk maken en vertalen in voor de opdrachtgever begrijpelijke termen (teruggrijpend op de testdoelen). Dit herhaalt zich totdat de opdrachtgever tevreden is over de balans tussen testzwaarte en -inspanning.

Tip

Een valkuil is dat de communicatie over de aangepaste strategie te "sterk" kan zijn.

Wanneer de opdrachtgever een aantal lichtere keuzes maakt, ontstaat een tabel met veel ○ -tjes. Wanneer deze tabel telkens weer getoond wordt in voortgangsverslagen of overleggen, geeft dit twee indrukken: 1) de opdrachtgever is roekeloos en 2) de testmanager gaat er niet helemaal voor en distantieert zich van de testaanpak. Om die reden is aan te raden deze tabelstijl enkel aan het begin en aan het einde van het testtraject te hanteren.

3. Fixeren testplan

Na de terugkoppeling en mogelijke aanpassing van het plan dient de testmanager het testplan ter goedkeuring voor te leggen aan minimaal de opdrachtgever. Wie verder goedkeuring moet geven, is afhankelijk van de organisatie. Bij veel organisaties wordt het testplan ook ter goedkeuring voorgelegd aan andere belanghebbenden, zoals gebruikers en ontwikkelaars. Partijen waaraan eisen zijn gesteld in de uitgangspunten van het plan, moeten hieraan hun goedkeuring geven.

Tips

- Om het proces van het tot stand komen van een testplan makkelijker te kunnen sturen en te voorkomen dat door kleine discussies het hele testplan steeds maar niet geaccordeerd wordt, kan er voor gekozen worden het testplan in delen te laten accorderen.
- De mate van formaliteit van de goedkeuring is afhankelijk van de organisatie. In sommige organisaties is het raadzaam de goedkeuring formeel te bekrachtigen door het laten tekenen van het testplan door de opdrachtgever en/of andere belanghebbenden. In andere organisaties kan worden volstaan met het sturen van een goedkeuring per mail of is een mondeling akkoord al voldoende.

Het plan wordt nu als formeel testproduct onder configuratiebeheer geplaatst. Daarnaast kan een presentatie, bijvoorbeeld aan de diverse betrokken partijen, bijdragen aan het verkrijgen van goedkeuring én, minstens zo belangrijk, draagvlak binnen de organisatie.

Producten

Het testplan.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.3 Fase Beheer

Doel

Het aan de opdrachtgever geven van voldoende inzicht in én sturingsmogelijkheden over:

- de voortgang van het testproces
- de kwaliteit en risico's van het testobject
- de kwaliteit van het testproces.

Hiervoor dient de testmanager het testproces op optimale wijze te beheersen en hierover te rapporteren.

Context

Deze activiteit heeft betrekking op de systeem- of acceptatietest. Het testen kan betrekking hebben op nieuwbouw, onderhoud, migratie of een pakketimplementatie of een mix, en de ontwikkelaanpak kan waterval, iteratief, agile of ook een mix zijn.

Randvoorwaarden

Deze activiteit start na het opstellen van het testplan.

Werkwijze

De testmanager en de beheerder(s)/infrastructuurcoördinatoren voeren de aan hen in het testplan toegekende activiteiten uit. Zij beheren het testproces, de infrastructuur en de testproducten. Op basis van deze gegevens analyseert de testmanager mogelijke trends. Tevens houdt de testmanager bijzonder goed voeling met ontwikkelingen buiten het testen, zoals vertraging bij de ontwikkelaars, komende grote wijzigingsvoorstellen en projectbijsturing. Indien nodig stelt de testmanager de opdrachtgever bepaalde sturingsmaatregelen voor.

Informatie is het belangrijkste product van testen. Hiertoe verzorgt de testmanager verschillende soorten rapportages aan de diverse doelgroepen, rekening houdend met de BDTM elementen Resultaat, Risico's, Tijd en Kosten.

Rollen/verantwoordelijkheden

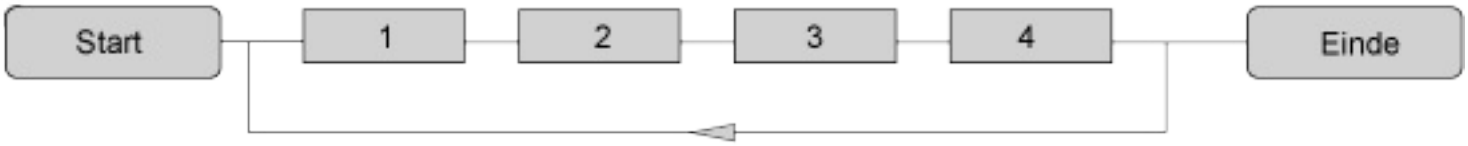
De primair verantwoordelijke rol voor het beheer van het testproces is de testmanager, soms ook testcoördinator genaamd.

Activiteiten

Het beheer van het testproces omvat de volgende activiteiten:

1. Uitvoeren beheer;
2. Bewaken;
3. Rapporteren;
4. Bijsturen.

Onderstaand schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten:



Figuur 6.11: Beheer van het testproces.

6.3.1 Uitvoeren beheer

Doel

Het beheren van het testproces, de bevindingen en de testproducten om voortdurend inzicht te kunnen bieden in de voortgang en kwaliteit van het testproces en de kwaliteit van het testobject.

Werkwijze

De beheeractiviteit kan in twee subactiviteiten worden onderscheiden:

- Volgens de in het testplan vastgestelde procedures worden de volgende vormen van beheer uitgevoerd: het beheer van het testproces, het testproductbeheer en het bevindingenbeheer (zie activiteit "Inrichten beheer", paragraaf 6.2.12). Het infrastructuurbeheer valt onder de fase "Inrichting en beheer Infrastructuur", zoals beschreven in paragraaf 6.4.
- Het testproces wordt ondersteund met én gecontroleerd op het toepassen van de normen en standaards voor beheer.

Deze werkzaamheden vallen onder de rol van beheerder of testmanager, zoals gedefinieerd in paragraaf 6.2.10 "Definiëren organisatie". In paragraaf 16.3 "Rollen niet als functie beschreven" zijn hiervoor de volgende beheerder-rollen onderkend: testprojectadministrateur, testwarebeheerder en bevindingenbeheerder.

Uitgediept

De echte uitdaging bij het beheer is niet zozeer het zelf volgen van de procedures, maar ervoor zorgen dat de overige testteamleden dat ook doen. Zaken als het indienen van urenbonnen, de testware onder configuratiebeheer plaatsen en bevindingen zorgvuldig administreren zijn niet altijd favoriete bezigheden. Maatregelen om te zorgen

dat het onder de aandacht blijft zijn:

- “Herhaling, herhaling, herhaling” van de boodschap dat goed beheer cruciaal is voor het slagen van het testproces. Maak het waarom en de voordelen duidelijk;
- Zet “Beheer” als onderwerp in het vaste teamoverleg;
- Spreek mensen er (direct) op aan wanneer zij zich niet of onvoldoende aan de afspraken houden.
- Controleer met name in het begin wanneer de testers de gewoonte nog moet aanwennen én bij aanvang van de testuitvoering wanneer de testers onder hoge tijdsdruk werken;

Let op: de testmanager kan tevens de rol van beheerder op zich nemen, maar dit ook uitbesteden aan een andere persoon. Het spreekt vanzelf dat in dat laatste geval deze beheerder wel de volledige support van de testmanager moet hebben wanneer de beheerder iemand aanspreekt op het niet nakomen van de procedures.

In de praktijk gaat het opstellen van normen en standaards vaak gelijk op met het ontwikkelen van de eerste producten en deze zijn dus nog niet aanwezig tijdens het schrijven van het testplan. De beheerder heeft dan tot taak om het ontwikkelen van de eerste producten te begeleiden en hier vervolgens algemeen geldende sjablonen van te maken.

Producten

Een beheerd testproces.

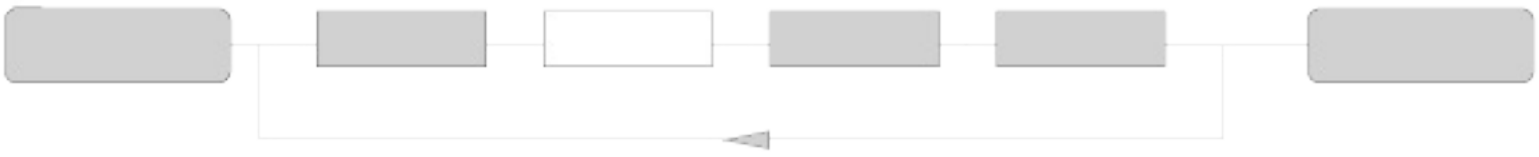
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

6.3.2 Bewaken



Doel

Het op basis van intern beheerde gegevens en externe informatie bewaken van het testproces.

Werkwijze

De voornaamste en moeilijkste taak van de testmanager is de bewaking van de uitvoering van het plan.

Hoewel dit in onderstaande paragraaf als een instrumentele activiteit is beschreven, is dit met name een *communicatieve activiteit*. Zo heeft de testmanager misschien wel het meeste werk aan het “bewaken” van de medewerkers in het team. Hieronder valt alles vanaf het aannemen van nieuwe testers gedurende het testtraject, het werk verdelen, werkoverleg/teamoverleg voeren, het ondersteunen, coachen en beoordelen van de medewerker tot en met het voeren van exit-gesprekken.

Een andere zeer belangrijke taak van de testmanager in ditzelfde verband is het onderhouden van de contacten met de wereld rondom het testteam, ook wel bekend onder namen als *stakeholder- en verwachtingsmanagement*. Kloppen de verwachtingen van de afnemers van het testen nog wel met wat het testen gaat opleveren? Zijn er ontwikkelingen in het project die van invloed zijn op het testproces?

Het mag duidelijk zijn dat voor bovengenoemde activiteiten een grote dosis sociale en communicatieve vaardigheden nodig zijn.

Algemeen gesteld moeten de in het plan beschreven activiteiten, zoals voorbereiden, specificeren en uitvoeren van de tests, volgens een bepaald tijdpad en in een bepaalde volgorde uitgevoerd worden. Hiervoor heeft de testmanager mensen beschikbaar (inclusief zichzelf). Voor een komende periode stelt de testmanager een detailplanning op wie wat gaat doen met hoeveel uren. De planning van het testplan is namelijk niet zo gedetailleerd dat tester A weet dat ze komende week testeenheden X en Z moet specificeren en tester B weet dat hij voor testeenheid Y testscripts Y1 en Y2 moet uitvoeren. De ervaring heeft geleerd dat een dergelijke detailplanning alleen werkt voor een eerstkomende korte periode, daarna blijken altijd veranderingen op te treden die een herplanning noodzakelijk maken. Voor de hand liggende periodes om een detailplanning op te stellen zijn de fasen: Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding. Bij iteratieve systeemontwikkeling maakt de testmanager ook per iteratie een detailplanning. Bij het opstellen van een detailplanning houdt de testmanager rekening met alle aspecten van planning zoals prioriteiten, beschikbaarheid en vaardigheden, zie ook paragraaf 6.2.7 “Bepalen planning”.

Een andere taak voor de testmanager is om “witte vlekken” uit het testplan in de loop van het testproces in te vullen. Dit is het geval wanneer bij het opstellen van het plan bepaalde informatie ontbreekt of er geen tijd is om een bepaalde (plan)activiteit uit te voeren.

Uitgediept

Als voorbeeld kan men denken aan het toewijzen van testeenheden en/of testtechnieken. Soms ontbreekt informatie van de ontwikkelaars om in het plan tot een goede opdeling in testeenheden te komen. Ook kan de testmanager besluiten om met de toewijzing van testtechnieken aan testeenheden te wachten tot de detailintake is uitgevoerd (zie paragraaf 6.5 “Fase Voorbereiding”).

Tegen het einde van de testuitvoering wordt de bewaking nog belangrijker omdat de testmanager dan de vraag moet kunnen beantwoorden of het verantwoord is om te stoppen met testen. De in het plan geformuleerde exit-criteria zijn hierbij leidend, maar wanneer deze er niet zijn of niet actueel meer zijn, zijn er toch enkele vuistregels te geven:

- Zijn alle (conform de laatste teststrategie) geplande tests doorlopen?
- Dit betreft nadrukkelijk niet de originele strategie, maar de laatst bijgestelde versie. Deze bevat de meest recente inzichten van opdrachtgever en testmanager over de balans tussen risico en testdekking.
- Zijn het aantal en de ernst van de nog openstaande bevindingen van een acceptabel niveau? En hieraan kan toegevoegd worden: zijn de kosten van het testen in deze periode hoger geworden dan de opbrengsten (“voorkomen schade”, zie onder)?
- Zijn het aantal nieuw gedane bevindingen én aantal opgeloste en geherteste bevindingen in de afgelopen periode (bijvoorbeeld week) minimaal geworden?
  Dit laatste punt zegt iets over de stabiliteit van het systeem. Soms wordt in de laatste periode zo veel hersteld en gehertest dat het systeem meerdere releases per dag kent. Wanneer alleen beide bovenste punten bekeken zouden worden, zou tot stoppen met testen worden besloten. Echter, het systeem is nog allesbehalve stabiel en een regressietest is zeer aan te bevelen.

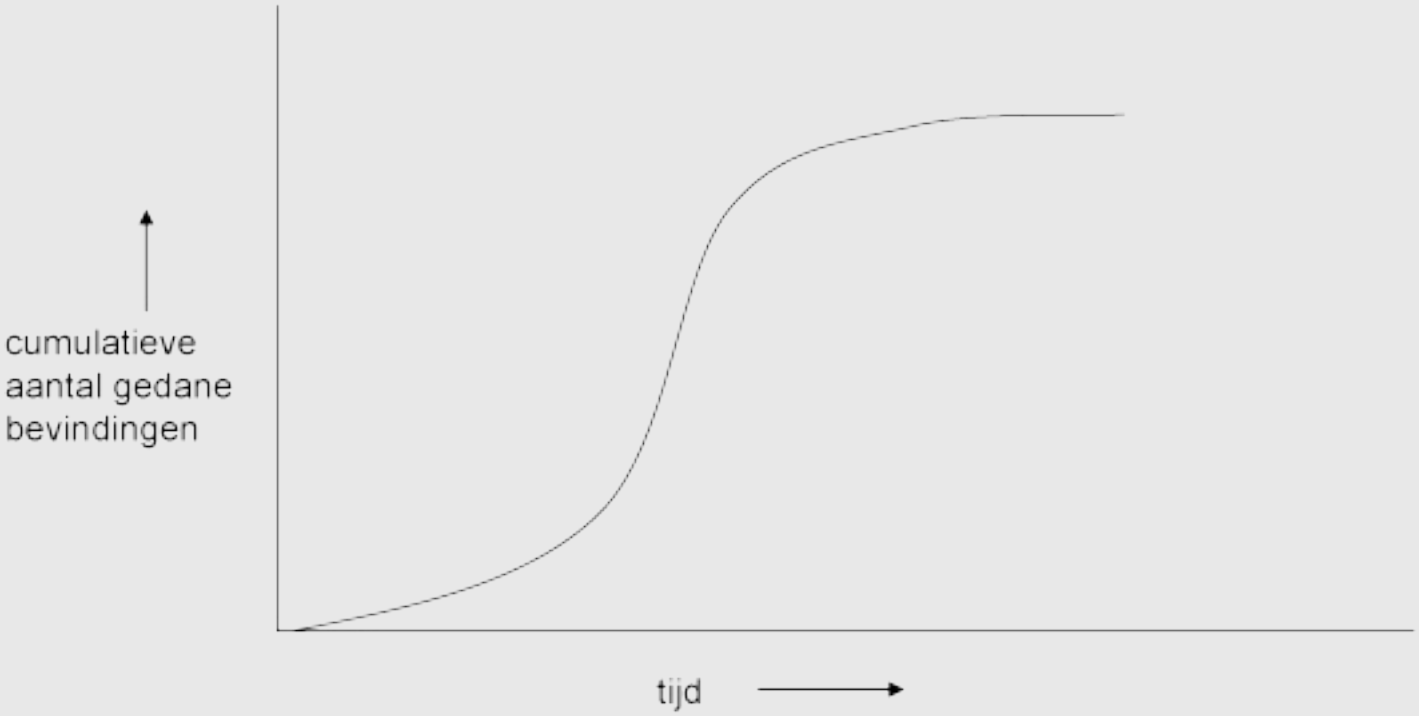
Pas wanneer op deze vragen een positief antwoord gegeven kan worden, is het verantwoord om het advies te geven om te stoppen met testen.

Nadat het testteam al haar taken heeft volbracht, inclusief de fase Afronding, vraagt de testmanager de opdrachtgever de opdracht te beëindigen en het testteam décharge te verlenen. Het team wordt dan ontbonden.

Tips

- Voorkomen schade: één van de baten van testen is dat er geen kosten ontstaan omdat een fout niet optreedt. Dit kan mogelijk benaderd worden door de ernst en oorzaak van een probleem te relateren aan eventuele

herstelkosten na in-productiename. Wat zijn de kosten wanneer de fout niet was gevonden?
- Het is mogelijk om een model te maken waarmee de voorkomen schade van elke bevinding geschat kan worden. Hierbij wordt aan aspecten van een bevinding (ernst, oorzaak en kwaliteitsattribuut) een bepaalde factor toegekend, bijvoorbeeld een ernstige bevinding geeft gemiddeld een 8 x zo hoge schade als een cosmetische bevinding. Door de voorkomen schade van een beperkt aantal bevindingen met hulp van experts in te schatten, worden de factoren bepaald en het gemiddelde bedrag per bevinding. Van een nieuwe bevinding kan dan snel de voorkomen schade worden ingeschat door het gemiddelde bedrag te vermenigvuldigen met de betreffende factoren, zie [Aalst, 1999].
- Een grafisch simpel maar handig plaatje is de S-curve van het gecumuleerde aantal gevonden fouten per dag. Als de S begint af te vlakken zou dit een indicatie kunnen zijn om te stoppen met testen. Dat is te bespreken met de betrokkenen.



Figuur 6.12: Voorbeeld S-curve.

Vervolgens leert de praktijk dat het oorspronkelijke plan aangepast zal gaan worden. De aanpassingen kunnen zowel een interne als externe oorzaak hebben, dat wil zeggen zowel binnen als buiten het testproces. Voor de testmanager is het zaak om deze gebeurtenissen of trends zo vroeg mogelijk te signaleren. Maatregelen om een negatieve trend bij te sturen kunnen dan tijdig worden genomen. Dit is vrijwel altijd beter, goedkoper en sneller.

Uitgediept

Hoewel er in de praktijk waarschijnlijk nog nooit een project is geweest waar het plan onveranderd is uitgevoerd, wil dat niet zeggen dat het plan daarmee onbelangrijk is. Integendeel, het plan biedt een gemeenschappelijk kader waardoor bijsturing makkelijker en effectiever mogelijk is dan wanneer er zonder plan gewerkt zou worden.

De informatie voor de gebeurtenissen of trends komt van de intern beheerde gegevens en van informatie van buiten het testproces, bijvoorbeeld notulen of memo's, maar zeker ook uit mondeling overleg als projectoverleg, standup meetings, bilaterale overleggen, enzovoort. Hier blijkt ook de waarde van een goed sociaal (project)netwerk voor de testmanager. Op basis van deze gegevens analyseert de testmanager mogelijke trends en probeert hij bedreigingen (of juist kansen) tijdig te signaleren. Gaat de trend doorzetten? Wat moet gebeuren om dit te voorkomen?

Hiertoe voert de testmanager de volgende stappen uit:

1. Analyseren gebeurtenis, inschatten risico's en definiëren maatregelen;
2. Afstemming met opdrachtgever en andere belanghebbenden (optioneel, afhankelijk van tolerantie).

1. Analyseren gebeurtenis,inschatten risico's en definiëren maatregelen

De testmanager analyseert de oorzaak van de gebeurtenissen en bepaalt de gevolgen voor het testproces. Daarbij kijkt hij expliciet naar de betekenis van de gebeurtenis voor de risico's die in het testtraject worden afgedekt.

Gebeurtenissen kunnen het testtraject voordelig maar ook nadelig beïnvloeden. Afhankelijk van het moment van optreden van de gebeurtenis, de analyse en de gevolgen voor het testproces bepaalt de testmanager mogelijke maatregelen.

Uitgediept

Onderstaand wat voorbeelden van veel voorkomende gebeurtenissen met oorzaken, gevolgen en maatregelen:

Gebeurtenis	***Het testobject wordt later opgeleverd dan gepland, terwijl de deadline voor het testproces gelijk blijft aan de oorspronkelijke datum***
Mogelijke oorzaken	De oorzaken hiervoor zullen veelal in het traject liggen vóór het testtraject. Mogelijkheden zijn hogere complexiteit dan verwacht, meningsverschillen of uitbreiding van de scope
Gevolgen voor het testproces	• Er is minder tijd voor het uitvoeren van de testgevallen • Er is meer tijd voor het specificeren van testgevallen • De benodigde middelen voor de testuitvoering, zoals bijvoorbeeld een representatieve testomgeving, hoeven pas later beschikbaar te zijn • …
Mogelijke maatregelen	• Er wordt extra testcapaciteit aangevraagd voor tijdens de testuitvoering • De testgevallen worden nog uitgebreider beschreven tijdens de specificatiefase waardoor ze eenvoudiger door onervaren testers zijn uit te voeren • Er worden testen parallel aan elkaar uitgevoerd • Rekening houdend met de risicoklasse wordt besloten om bepaalde testactiviteiten minder of niet uit te voeren • Er wordt besloten om bepaalde testsoorten of testvormen in elkaar te schuiven waardoor ze gezamenlijk uitgevoerd worden • Dakpanmethode m.b.t. specificatie testgevallen en uitvoering, ofwel vaker kleinere opleveringen doen van ontwikkeling aan testen • Aanvullend budget vragen • …

Gebeurtenis	***De productiviteit van de medewerkers is lager dan verwacht***
Mogelijke oorzaken	• De kwaliteit van de testbasis is minder dan vooraf ingeschat • De medewerkers hebben gemiddeld minder ervaring dan gebruikt is voor het vaststellen van de productiviteit • Het testobject is complexer dan vooraf ingeschat waardoor het opstellen van testgevallen moeilijker is • …
Gevolgen voor het testproces	De testactiviteiten duren langer dan vooraf ingepland
Mogelijke maatregelen	• Medewerkers worden vervangen door andere medewerkers die meer ervaring hebben • Er wordt extra capaciteit aangevraagd bij de opdrachtgever • De testwerkwijze wordt aangepast waardoor minder risicovolle delen nog minder aandacht krijgen en er daardoor meer tijd beschikbaar is voor de risicovolle onderdelen • Er wordt besloten om niet meer alle testgevallen uitgebreid uit te werken en bijvoorbeeld alleen logisch te beschrijven (dit stelt in het algemeen hogere eisen aan de testuitvoerder) • …

Gebeurtenis	***Het specificeren van testgevallen kost meer tijd dan gepland***
Mogelijke oorzaken	• De testbasis is van minder kwaliteit dan vooraf ingepland • De productiviteit van de medewerkers is lager dan vooraf ingeschat • …
Gevolgen voor het testproces	Uitloop van testspecificatie kan betekenen dat niet op tijd met de testuitvoering gestart kan worden
Mogelijke maatregelen	• Er worden technieken gekozen die leiden tot minder testgevallen. Dit betekent overigens dat er sprake is van minder testdekking en dat de onderkende risico's minder afgedekt worden • De logische testgevallen worden niet uitgeschreven in fysieke testgevallen (dit stelt in het algemeen hogere eisen aan de testuitvoerder) • Er wordt extra tijd of capaciteit gevraagd aan de opdrachtgever • Extra ondersteuning door materiedeskundigen of ontwikkelaars • …

Gebeurtenis	***De testbasis verandert steeds***

Mogelijke oorzaken	• De testbasis is van minder kwaliteit dan vooraf ingepland • De scope van het project wordt steeds groter • Er zijn meningsverschillen in het project over de te leveren functionaliteit • …
Gevolgen voor het testproces	• (Tijdens specificatie en uitvoering) De testspecificaties moeten steeds herzien worden en zijn nooit af • (Tijdens uitvoering) Er zijn continu extra hertests nodig
Mogelijke maatregelen	• De logische testgevallen worden niet uitgeschreven in fysieke testgevallen (dit stelt in het algemeen hogere eisen aan de testuitvoerder). • Exploratory testing als techniek om minder afhankelijk te zijn van de testbasis én deze zo laat mogelijk nodig te hebben • Er wordt extra tijd of capaciteit gevraagd aan de opdrachtgever. • Strenger configuratie- en changemanagament op projectniveau, (met vanzelfsprekend betrokkenheid van testen) • …

Uitgediept

Hertesten

Een specifiek onderdeel van de strategie is hoe om te gaan met hertesten. Normaal gesproken levert een test bevindingen op die daarna hersteld worden. Voor de hertest moet dan een keuze gemaakt worden. Er kan bijvoorbeeld een beperkte hertest worden uitgevoerd die alleen gericht is op het testen van de aanpassing. Een andere mogelijkheid is om een hertest uit te voeren voor de totale functie waarin de aanpassing is doorgevoerd, voor de totale functie in samenhang met omliggende functies of zelfs voor het totale systeem. Ook kan de wijziging gehertest worden met specifieke testgevallen en kan op de (ongewijzigde) rest van het systeem een regressietest gedraaid worden. De keuze voor de diepgang van de hertest is gebaseerd op de risico's. Soms zijn er richtlijnen opgesteld, soms bepaalt de testmanager van geval tot geval de gewenste hertestdiepgang. Feitelijk voert de testmanager dan een soort mini-teststrategiebepaling uit, waarbij alle stappen kort worden doorlopen.

2. Afstemming met opdrachtgever en andere belanghebbenden (optioneel, afhankelijk van tolerantie)

Afhankelijk van de uit te voeren maatregelen kan de testmanager deze zelfstandig doorvoeren of is vooraf afstemming met de opdrachtgever en eventueel andere belanghebbenden noodzakelijk. De vorm van deze afstemming is organisatieafhankelijk. In de praktijk wordt hierbij vaak gebruik gemaakt van rapportages. De verschillende rapportagevormen zijn beschreven in paragraaf 6.3.3 "Rapporteren".

De ruimte die de testmanager heeft om zelfstandig maatregelen te nemen wordt bepaald door de volgende factoren:

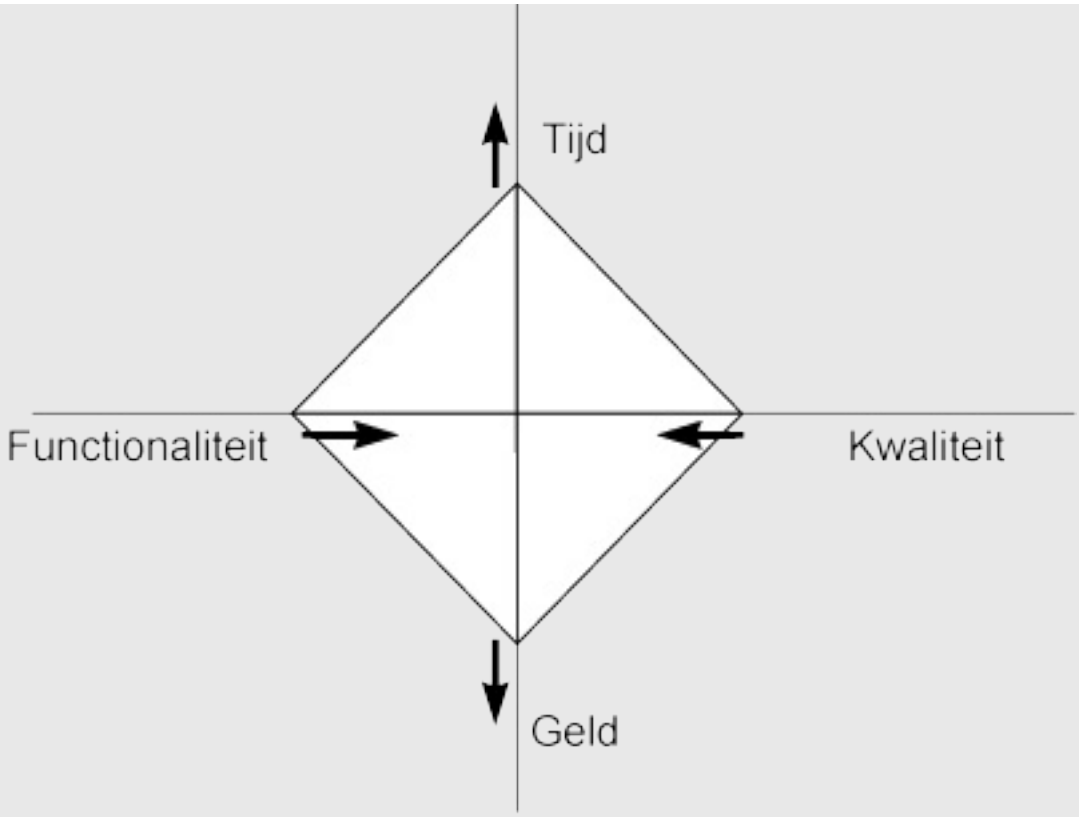
- De mate waarin de problemen voor het testproces kunnen worden opgelost binnen de gestelde opdracht, productrisicoanalyse, teststrategie, begroting, planning en overige randvoorwaarden. Met andere woorden: de opdrachtgever merkt er eigenlijk niets van.
- De mate waarin de problemen kunnen worden opgelost binnen de tolerantiegrenzen, die in de planfase (paragraaf 6.2.7 "Bepalen planning") zijn afgesproken.

In de praktijk moet de testmanager in het algemeen minimaal toestemming vragen als de maatregelen van invloed zijn op de afspraken zoals die in de planfase zijn gemaakt. Dit wil zeggen als er aanpassingen moeten worden doorgevoerd in de opdrachtformulering, productrisicoanalyse, teststrategie, begroting en/of planning.

Uitgediept

Duivelsvierkant

Een bekende trend is gesymboliseerd in het Duivelsvierkant, met Tijd, Geld, Functionaliteit en Kwaliteit als hoekpunten. Bij aanvang van een project is tussen de hoekpunten een zeker evenwicht. Een voorspelbaar projectverloop is vervolgens dat allerlei onvoorziene gebeurtenissen optreden die het vierkant onder spanning zetten (zie figuur 6.13). Met name lopen bepaalde activiteiten uit (Tijd) en/of kosten veel meer dan gepland (Geld). De projectmanager stuurt dit nog al eens bij door te beperken op de andere hoekpunten, Kwaliteit en Functionaliteit.



Figuur 6.13: Duivelsvierkant.

Hoewel dit op zich geen “verkeerd” gedrag van de projectmanager hoeft te zijn, is het de taak van de testmanager om Functionaliteit en Kwaliteit te bewaken. De testmanager geeft, met het vierkant in het achterhoofd, de consequenties aan van de beslissingen van de projectmanager en signaleert bijvoorbeeld naar de opdrachtgever wanneer de keuzes iedere keer vallen op het beperken van Kwaliteit en Functionaliteit. Met name het tijdig signaleren van deze trend is moeilijk en benadrukt het belang van een onafhankelijke testmanager. Afhankelijk van hoe men kijkt, is de testmanager het “geweten” of de “luis in de pels” van de projectmanager en/of opdrachtgever. Deze rol vraagt een hoge mate van professionaliteit omdat de testmanager zorgvuldig om moet gaan met het politieke spel tussen de diverse belangen binnen en buiten het project.

Tips

- Een tip die in bovenstaand verband gegeven kan worden, is om bij wijzigingsverzoeken of bij door de projectmanager voorgestelde bijstellingen van het testen nooit gelijk de hakken in het zand te zetten en “nee, onmogelijk want …” te roepen. Beter is het om te reageren in de trant van “Hmm, interessant idee. Laten we dat eens verder uitwerken, wat zou dat betekenen voor …. Het idee zou kunnen werken als we met z’n allen die en die consequenties accepteren.”
- In de fasen Voorbereiding en Specificatie tendeert het (project)management naar onverschilligheid ten aanzien van het testen. Pas bij de testuitvoering, op het moment dat het testen op het kritieke pad zit, is men geïnteresseerd in de voortgang. De testmanager moet hier rekening mee houden door bijvoorbeeld de vorm en frequentie van rapportage en overleg (meer en vaker bij testuitvoering) aan te passen, maar kan ook zijn eigen verwachting hierop instellen (“van figurant tot ster”).

Producten

Voorgestelde sturingsmaatregelen.

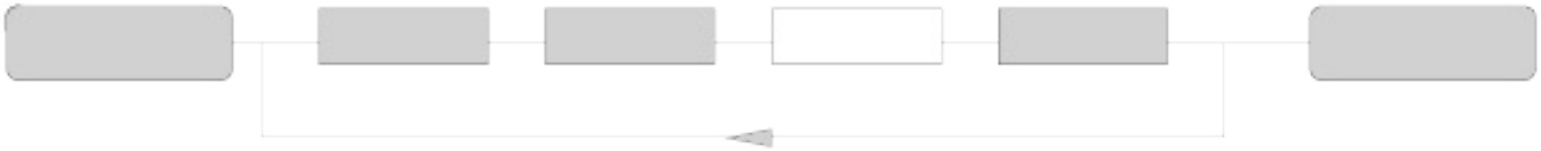
Technieken

Niet van toepassing.

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.
Workflowtool.

6.3.3 Rapporteren



Doel

Het opstellen van rapportages om inzicht te verschaffen in zowel de kwaliteit van het testobject als de voortgang en kwaliteit van het testproces. Deze rapportages zorgen ervoor dat de opdrachtgever en andere belanghebbenden effectief kunnen sturen op het verloop van het testtraject.

Werkwijze

Gedurende het testproces stelt de testmanager verschillende rapportages op.

In het testplan zijn in het hoofdstuk “Beheer” de vorm en de frequentie van rapportage vastgesteld. Periodiek wordt gerapporteerd over de kwaliteit van het testobject en de voortgang en kwaliteit van het testproces. Naast de periodieke rapportages, kan door de opdrachtgever of andere betrokkenen worden gevraagd om ad-hoc rapportages. Het bekendste voorbeeld hiervan is het risicorapport, om de mogelijke gevolgen van een bedreiging of risico voor het testtraject in kaart te brengen en maatregelen voor te stellen. Daarnaast kan ook opeens gevraagd worden om een extra voortgangsrapport, bijvoorbeeld als meest actuele input voor een stuurgroep- of projectmanagementoverleg. Aan het einde van het testproces worden een vrijgaveadvies en eindrapport gemaakt.

Met al deze informatie krijgen opdrachtgever, projectmanager en andere betrokkenen inzicht in de mate waarin:

- het beoogde resultaat is behaald;
- de risico’s bij in productie name bekend en zo klein mogelijk zijn binnen de gestelde randvoorwaarden;
- dit binnen het vastgestelde budget en de vastgestelde (doorloop)tijd gebeurt.

Ofwel, dit zijn de BDTM aspecten Resultaat, Risico’s, Tijd en Kosten. Het geven van inzicht betekent dat het rapport moet aansluiten op de belevingswereld van de ontvanger(s) ervan.

De rapportages zijn gebaseerd op de gegevens zoals vastgelegd volgens paragraaf 6.2.12 “Inrichten Beheer”.

Tips

- Rapporteer altijd juist en volledig, want niemand is er bij gebaat de zaken mooier voor te stellen dan ze zijn;
- Rapporteer secuur en zorg voor goede cijfermatige onderbouwing;
- Rapporteer in de terminologie van de opdrachtgever, niet alleen in aantallen bevindingen;
- Rapporteer ook positief nieuws, bijvoorbeeld het aantal testgevallen dat zonder bevindingen is doorlopen;
- Sluit qua detailniveau in de rapportage aan op de behoefte van de doelgroep;
- Wees neutraal in de bewoordingen, word niet persoonlijk;
- Antwoord op vragen als “Kan ik in productie?” of “Kan het geaccepteerd worden?” liever niet met “Nee!”, maar met “Ja, mits … “ of “Nee, tenzij …”.

Onderstaand wordt de inhoud van de belangrijkste rapportages beschreven:

a. voortgangsrapport
b. risicorapport
c. vrijgaveadvies
d. eindrapport

a. Voortgangsrapport

Conform de in het testplan beschreven rapportagestructuur wordt gerapporteerd. De voortgangsrapportage bevat gegevens over de meest recente rapportageperiode en gecumuleerde gegevens over het gehele testproces.

Naast cijfermateriaal moet het rapport ook tekstuele toelichting en advies geven over de resultaten, voortgang, risico's en eventuele knelpunten. Zeker bij rapportages die gegenereerd worden uit voortgangsbewakingstools wordt dit laatste nogal eens vergeten. Beseft moet worden dat de toelichting en het advies erg belangrijk zijn om snel en goed inzicht te krijgen in de cijfers. Het is het belangrijkste product van testen! Hoewel de toelichting ook mondeling gegeven kan én moet worden, dient deze zeker ook schriftelijk in het rapport te staan. Dit dwingt de testmanager om er vooraf goed over na te denken, maakt het advies sterker, bereikt een groter publiek en helpt bij procesevaluatie achteraf.

Uitgediept

Voortgangsrapport versus eindrapport

Hoewel de termen tussen- of voortgangsrapport kunnen suggereren dat deze minder belangrijk zijn dan het eindrapport, is het in werkelijkheid precies andersom. Het voortgangsrapport geeft vroege informatie en advies, waarmee de ontvangers (zoals opdrachtgever, projectmanager en anderen) het totale project of proces vaak nog tijdig kunnen bijsturen om het totaal in goede banen te leiden. Het eindrapport is meer een achteraf-evaluatie waar met name volgende testprocessen en projecten van kunnen profiteren.

Een voortgangsrapport kent op hoofdlijnen de volgende inhoud. Deze is gebaseerd op de BDTM-visie met de vier aspecten Resultaat, Risico's, Tijd en Kosten. In de praktijk kan de inhoudsopgave overigens een andere volgorde kennen, kunnen onderwerpen gecombineerd zijn of zelfs weggelaten. Dit hangt af van de doelgroep van het rapport.

1. Status van het testobject (BDTM: resultaat)
   1.2. Status per kenmerk/deelobject
   1.2. Status testdoelen
   1.3. Trends en aanbevelingen
2. Productrisico- en strategiebijstelling (BDTM: risico's)
3. Voortgang van het testproces (BDTM: tijd en kosten)
   3.1. Voortgang (in uren en data) activiteiten of producten over de afgelopen periode
   3.2. Activiteiten komende periode
   3.3. Verliesuren
   3.4. Trends en aanbevelingen
4. Knelpunten/bespreekpunten (alle BDTM-aspecten)
5. Afspraken
6. Kwaliteit van het testproces (optioneel, alle BDTM aspecten)
   6.1. Effectiviteit
   6.2. Efficiëntie
   6.3. Controleerbaarheid

Deze onderwerpen worden onderstaand nader toegelicht.

1. Status van het testobject (resultaat)

1.1 Status per kenmerk/deelobject

Per kenmerk/deelobject wordt weergegeven:

- De status van de tests (niet gestart, plan, specificatie, uitvoering, hertest X, afgerond), optioneel met het percentage voortgang, bijvoorbeeld dat de voortgang van uitvoering op 60% geschat wordt;

- Overzicht van aantallen bevindingen (gesorteerd op status en ernst, optioneel ook op andere velden zoals oorzaak);
- wanneer testproducten (zoals testgevallen of -scripts) nadrukkelijk als resultaat worden gezien, kunnen deze ook in het overzicht worden opgenomen, met indicatie of al begonnen is aan het product en of het klaar is.

Naarmate het einde van de testperiode dichterbij komt, wordt in de voortgangsrapportage steeds meer aandacht besteed aan de consequenties van openstaande bevindingen. In het begin is dit minder zinvol om in het rapport op te nemen, omdat verwacht mag worden dat de bevindingen opgelost worden. Wél dienen de consequenties altijd in het bevindingrapport zelf opgenomen te worden.

- Nog openstaande bevindingen en hun impact;
- Niet opgeloste bevindingen (known errors), en hun impact.

1.2 Status testdoelen

Op basis van bovenstaande moet de status per testdoel (user requirement, bedrijfsproces, kritische succesfactor, enzovoort) worden gerapporteerd. Soms is een testdoel rechtstreeks te koppelen aan een aantal kenmerken/deelobjecten en aan de daaraan gerelateerde teststatus, soms is de hierboven beschreven status per kenmerk/deelobject onvoldoende bruikbaar en moet de testmanager per testdoel alsnog de teststatus bepalen. De detail PRA-tabellen uit de Productrisicoanalyse maken de koppeling mogelijk.

1.3 Trends en aanbevelingen

Relevante trends en daaraan gerelateerde aanbevelingen kunnen hier gerapporteerd worden.

Uitgediept

Onderstaand een aantal overzichten die duidelijk maken of bepaalde trends spelen:

- Aantal openstaande bevindingen per week geeft aan of het testen naar een eind kan lopen of dat er een boeggolf van achterstallig werk aan het ontstaan is;
- De verhouding tussen aantallen bevindingen en testgevallen per deelsysteem geeft een indicatie of extra testen op dat onderdeel nog veel meer bevindingen gaat opleveren;
- Aantal gedane bevindingen en aantal opgeloste (inclusief geherteste) bevindingen in een bepaalde periode zegt iets over de stabiliteit van het systeem;
- Status bevinding versus wie de volgende stap moet uitvoeren zetten in de afhandeling ervan. Dit geeft aan waar bottleneck ligt. Bijvoorbeeld alle complexe fouten worden toegewezen aan die ervaren ontwikkelaar waar vervolgens een stuwmeer van op te lossen bevindingen ontstaat;
- Oorzaak bevindingen (requirements, bouw, testomgeving, foutief installeren/draaien, fout testgeval) versus deelsysteem. Geeft inzicht in de concentraties van specifieke foutoorzaken;
- Aantal bevindingen versus tabellen (bij datawarehousing). Dit vertelt wat de foutgevoelige systeemonderdelen zijn.

Uitgediept

Om de trend voor de belanghebbenden aansprekend te maken, is het aan te bevelen om hier gebruik te maken van grafieken, waarmee de trend zichtbaar wordt gemaakt. Dit is minder makkelijk dan het lijkt. Een goed leesbare en duidelijke grafiek maken is moeilijk. Aanwijzingen zijn (vrij naar [Tufte, 2001):

1. Zet de gegevens en de boodschap centraal
2. Maximaliseer de data/inkt verhouding (ofwel laat alle tekens, lijntjes, kleurtjes, die niets toevoegen weg)
3. Verwijder onnodige redundantie
4. Herzie en pas aan.

2. Productrisico- en strategiebijstelling (risico's)

In dit onderdeel van de rapportage krijgen de belanghebbenden inzicht in de mate waarin de afdekking van de

verschillende productrisico's is veranderd én in eventuele procesrisico's.

In de teststrategie van het testplan is bepaald of en in welke mate productrisico's met testen worden afgedekt. Gedurende het testproces gaan hierin mogelijk afwijkingen optreden: de inschatting van risico's blijkt anders en/of de testafdekking moet bijgestuurd worden. De bijstelling over de rapportageperiode, met bijbehorende consequenties, wordt in dit onderdeel gerapporteerd. Hierbij wordt de vertaling gemaakt naar de testdoelen: wat hebben de veranderde risico's voor impact op het halen van deze doelen?

3. Voortgang van het testproces (tijd en kosten)

Ten aanzien van de voortgang van het testproces zijn de onderstaande gegevens van belang.

3.1 Voortgang over de afgelopen periode

Op het niveau van fasen en/of hoofdproducten kan het volgende worden gerapporteerd:

- Aantal geplande uren;
- Aantal bestede uren tot nu toe;
- Aantal uren dat naar verwachting nog besteed moet worden;
- Percentage afgerond;
- Datums: geplande/verwachte/feitelijke start, geplande/verwachte/feitelijke eind;

Producten kunnen zijn testplan, testscripts, testuitvoeringsdossiers en rapportages.

Als de testmanager ook budgetverantwoordelijk is, neemt hij ook gegevens over het binnen budget afronden van het testproces op in de voortgangsrapportage.

3.2 Activiteiten komende periode

Hier worden de in de komende periode uit te voeren activiteiten gerapporteerd.

3.3 Verliesuren

Dit zijn niet-productieve uren van de testers. Als de omgeving van het testproces niet aan bepaalde randvoorwaarden voldoet, levert dit inefficiëntie en verlies van uren op. Voorbeelden zijn een niet-functionerende testinfrastructuur, veel of langdurige testblokkerende bevindingen of gebrek aan ondersteuning. Verliesuren met oorzaak worden hier gerapporteerd.

3.4 Trends en aanbevelingen

Net als bij de trends voor de status van het testobject moeten ook trends en aanbevelingen rondom de testvoortgang gerapporteerd worden. De centrale vraag hierbij is of de afgesproken mijlpalen haalbaar (lijken te) zijn.

Uitgediept

Een trend waarop gelet kan worden, is wat de gemiddelde benodigde tijd voor herstel van een bevinding is. Loopt dit op, dan is dit mogelijk een signaal dat de hoeveelheid achterstallig werk fors aan het groeien is. Zo kan ook het percentage verkeerd herstelde bevindingen in de gaten gehouden worden.

4. Knelpunten/bespreekpunten (alle BDTM-aspecten)

In deze paragraaf van het rapport geeft de testmanager eventuele knelpunten of bespreekpunten aan die de afronding van de testopdracht binnen de gestelde beperkingen van tijd en kosten in gevaar brengen. Denk hierbij bijvoorbeeld aan:

- Het testobject wordt later opgeleverd dan gepland;
- De testbasis is van mindere kwaliteit dan verwacht;
- De testomgeving is niet beschikbaar op de afgesproken datum;

- Op de testomgeving of het testobject zijn testblokkerende bevindingen.

Naast de verschillende knelpunten worden ook de consequenties hiervan en de mogelijke maatregelen weergegeven. Ook hier maakt de testmanager de vertaling naar de testdoelen.

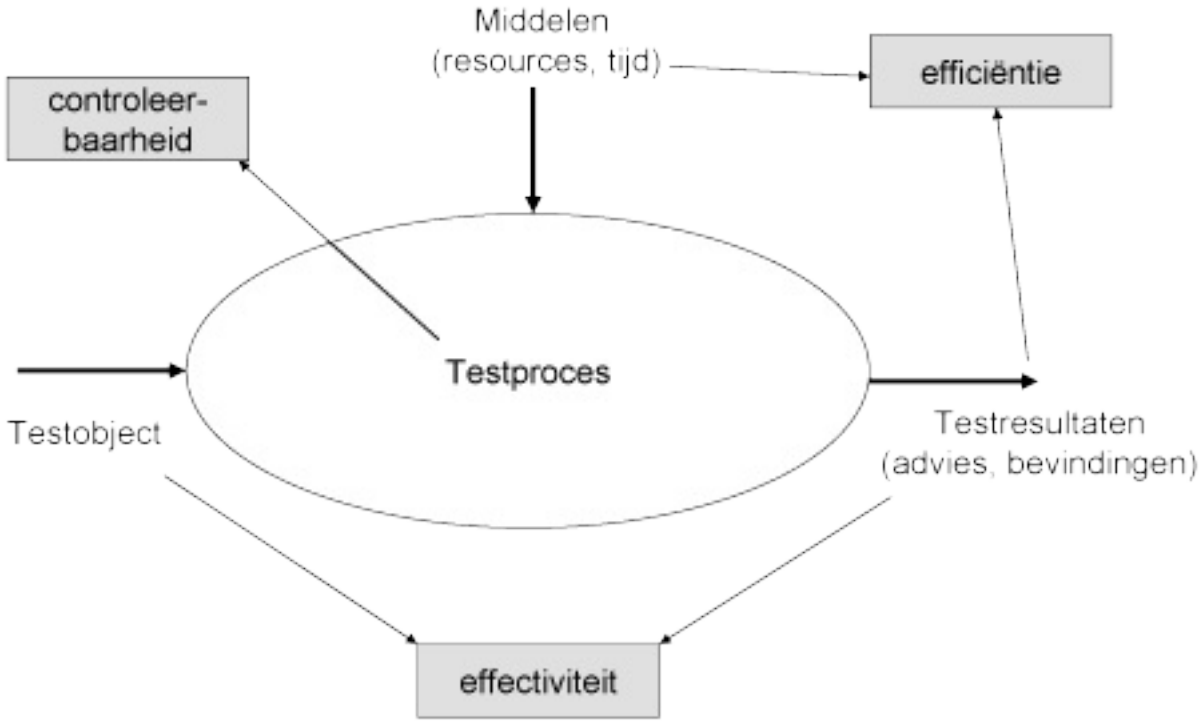
5. Afspraken

Hier staan de in lopende verslagperiode gemaakte afspraken tussen het testteam en andere partijen die relevant zijn voor de ontvangers van het rapport.

6. Kwaliteit van het testproces (optioneel)

Indien gewenst, kan in dit deel van het rapport informatie opgenomen worden met betrekking tot de kwaliteit van het testproces. Ten aanzien van de kwaliteit van het testproces spelen de volgende vragen:

- zijn de belangrijke fouten (zo vroeg mogelijk) gevonden? (effectiviteit)
- hoe zuinig gaat het testproces om met tijd en middelen? (efficiëntie)
- werkt het testproces zoals is afgesproken? (controleerbaarheid)



Figuur 6.14: De drie kwaliteitsaspecten van het testproces.

Uitgediept

Aandachtspunt hierbij is het algemene probleem met metrics: hoe trekt men de goede conclusies uit de cijfers, hoe kan men voorkomen dat appels met peren worden vergeleken. Zie hiervoor ook hoofdstuk 13 “Metrics”.

6.1 Effectiviteit

Uitgediept

Het lastige van de vraag of het testen effectief is, is dat dit meestal pas achteraf vastgesteld kan worden. De effectiviteitvraag kan in tweeën gesplitst worden:

- Is er een goede strategie?
- Is er getest conform deze strategie?

Er zijn diverse indicatoren te geven voor opname in het rapport:

- het percentage gevonden fouten in de testsoort / (benadering van) het aantal aanwezige fouten, dit laatste kan benaderd worden door bijvoorbeeld het aantal fouten te tellen dat tijdens eerste 3 maanden productie nog gevonden wordt;
- het percentage gevonden fouten in een testsoort die redelijkerwijs al in een voorgaande test gevonden hadden moeten worden (30% van de in de acceptatietest gedane bevindingen betreffen programmafouten; deze hadden eigenlijk al in de ontwikkeltests en systeemtest gevonden moeten worden;
- dekkingsgraad van testen; des te zwaarder de test, des te meer gevonden gaat worden;
- het percentage onterechte bevindingen (= testfouten).

6.2 Efficiëntie

Uitgediept

Hiervoor zijn de volgende indicatoren mogelijk:

- het aantal bevindingen / testuur;
- (door het vinden van fouten) inschatting van voorkomen schade ten opzichte van de testkosten;
- aantal gespecificeerde of uitgevoerde testgevallen per uur;
- aantal gereviewde pagina's per uur.

Door deze cijfers te vergelijken met een vastgestelde norm ontstaat een beeld van de efficiëntie van het testproces.

6.3 Controleerbaarheid

Uitgediept

Dit aspect is moeilijk te vangen in indicatoren. Wat de testmanager hierover in het rapport kan zetten is of en hoe de afgelopen periode is gecontroleerd dat het testteam werkt zoals afgesproken. De controle kan zich richten op de testproducten of de processen en kan op basis van de geplande kwaliteitsmaatregelen, of door monitoring of een steekproef op overkoepelend niveau. De testmanager dient een goede risicoinschatting te maken óf en wát zinvol is te controleren. Met name de testsoorten met onervaren testmanagers of die geoutsourced zijn, komen in aanmerking.

Onderstaand een voorbeeld van een dashboard waarmee in één oogopslag de belangrijkste informatie wordt weergegeven.

Onderdeel	Nu	Vorig	Opmerking
1. Kwaliteit testobject	☹	😐	…
2. Risico's	😐	😐	…
3. Voortgang	☺	☺	…
4. Kwaliteit testproces	☺	😐	…

In het verdere rapport worden deze punten dan verder in detail uitgewerkt in overzichten met toelichting. Voorbeelden van overzichten (zonder toelichting):

Kwaliteit testobject – bevindingen

	Openstaand	Te hertesten	Afgesloten	TOTAAL

Blokkerend			4	**4**
Ernstig	1	1	12	**14**
Storend	3	12	49	**64**
Cosmetisch	15	7	37	**59**
TOTAAL	**19**	**20**	**102**	**141**

Kwaliteit testobject – deelsysteem x oorzaken

	Requirements	**Ontwerp**	**Software**	**Infrastructuur**	**Test**	**TOTAAL**
Deelsysteem 1	3	5	18	6	2	**34**
Deelsysteem 2		1	16	2	4	**23**
Deelsysteem 3	6	14	30	14	1	**65**
Totale systeem						
TOTAAL	**9**	**20**	**64**	**22**	**7**	**122**

Voortgang

	Tijd			**Uren**			
Mijlpaal/Activiteit	**Plan**	**Verwacht**	**Gerealiseerd**	**Begroot (A)**	**Besteed (B)**	**Nog te besteden (C)**	**Verschil (A-B-C)**
Planning							
- Testplan	1 apr		1 apr	60	54	0	6
Voorbereiding							
- Detailintake	8 apr	12 apr		40	46	4	-10
Specificatie							
- Testeenheid	2 mei	2 mei		120	60	60	0
…							

b. Risicorapportage

De doelstelling van de risicorapportage is de verschillende belanghebbenden voldoende informatie te geven om onderbouwde beslissingen te nemen ten aanzien van het testproces. De informatie in de risicorapportage moet dan ook gericht zijn op de gevolgen van de gebeurtenis voor het behalen van het afgesproken resultaat binnen de afgesproken (doorloop)tijd en kosten.

Uitgediept

De testmanager stelt een risicorapportage op als er gebeurtenissen optreden waarvoor maatregelen genomen moeten worden waarvoor de testmanager niet beslissingsbevoegd is.

Een andere reden om een risicorapportage op te stellen is als de opdrachtgever aan de testmanager vraagt om consequenties en mogelijke maatregelen op te stellen voor een of meer scenario's waarover een beslissing genomen moet worden. Denk hierbij bijvoorbeeld aan het scenario dat de opdrachtgever merkt dat de realisatie uitloopt en hij/zij overweegt om hiervoor budget van het testtraject ter beschikking te stellen.

In een risicorapportage worden minimaal de volgende onderwerpen behandeld:

- Beschrijving van de gebeurtenis / het scenario
- Consequenties van het optreden van de gebeurtenis voor het testtraject.
  Met andere woorden wat zijn de gevolgen?
- Betekenis van het optreden van de gebeurtenis voor de mate waarin de verschillende productrisico's worden afgedekt
- Mogelijke maatregelen
  Indien mogelijk schetst de testmanager hierbij meerdere maatregelen met bijbehorende kosten. Bovendien wordt een inschatting gemaakt van de invloed van de maatregelen op de onderkende consequenties en mate van afdekken van productrisico's.
- Advies
  De testmanager geeft een advies ten aanzien van de te kiezen maatregel(en).

c. Vrijgaveadvies

Het vrijgaveadvies wordt aan het einde van de testuitvoering opgesteld. Doelstelling van het vrijgaveadvies is om de opdrachtgever en andere betrokkenen een zodanig inzicht te geven in de kwaliteit van het testobject dat zij onderbouwd

kunnen beslissen of het testobject in de huidige status door kan naar de volgende fase. Met volgende fase wordt in dit verband een volgende testsoort of productie genoemd. Om die reden wordt het vrijgaveadvies meestal onder hoge tijdsdruk opgesteld, want vlak na uitvoering van de laatste tests en vóórdat het testobject aan de volgende fase wordt vrijgegeven, is meestal maar héél weinig tijd beschikbaar. De testmanager doet er daarom goed aan om tegen het einde van de laatste tests het concept van het vrijgaveadvies al grotendeels klaar te hebben, zodat alleen nog de laatste testresultaten verwerkt hoeven te worden.

De informatie in het vrijgaveadvies mag eigenlijk geen verrassing zijn voor de opdrachtgever. Door tijdens het traject goede voortgangsrapportages en indien nodig risicorapportages op te stellen, is de opdrachtgever steeds op de hoogte van de voor hem relevante informatie.

Om de opdrachtgever van de benodigde informatie te voorzien, moet het vrijgaveadvies minimaal ingaan op de volgende onderwerpen:

- Advies of het vanuit testoptiek verantwoord is om het testobject in de huidige status over te dragen aan de volgende fase
  De uiteindelijke beslissing om al dan niet naar de volgende fase te gaan, ligt buiten het testproces. Hierin spelen namelijk veel meer factoren een rol dan de factoren waarop het testproces betrekking heeft. Denk hierbij bijvoorbeeld aan politieke of commerciële belangen waardoor een overgang naar een volgende fase ondanks een negatief vrijgaveadvies niet uitgesteld kan worden.
- Wel en niet behaalde resultaten
  Welke testdoelen zijn gehaald en welke niet of slechts in bepaalde mate? Op basis van de testresultaten van de kenmerken en deelobjecten geeft de testmanager een oordeel en advies over de door de opdrachtgever gestelde testdoelen. Ook wordt aangegeven of de exit-criteria zijn gehaald. Het aantal en de zwaarte van de openstaande bevindingen spelen hier een belangrijke rol. Per bevinding wordt aangegeven wat de consequenties van die bevinding voor de organisatie zijn. Indien mogelijk worden ook risicobeperkende maatregelen aangegeven. Hierbij kan bijvoorbeeld worden gedacht aan een work around waarmee het testobject naar de volgende fase kan zonder dat de bevinding is opgelost.
- Risico-inschatting
  Aan het begin van het testproces is tijdens de planfase met de opdrachtgever overeen gekomen in welke mate productrisico's met welke zwaarte van testen zouden worden afgedekt. Om diverse redenen kan er voor gekozen zijn bepaalde onderdelen lichter af te dekken met testen dan de risico-inschatting aangeeft. Daarnaast zijn tijdens het testproces meestal nog allerlei veranderingen aangebracht op de oorspronkelijke strategie én is de oorspronkelijke risico-inschatting mogelijk bijgesteld, mogelijk resulterend in meer of andere risico's. In dit onderdeel van het vrijgaveadvies signaleert de testmanager welke kenmerken of deelobjecten lichter (of niet) zijn getest dan de risico's rechtvaardigen en daarmee een hoger risico geven. Bovendien worden de bijbehorende consequenties weergegeven.

d. Eindrapport

Doelstelling van dit rapport is het verkrijgen van inzicht in de wijze waarop het testproces is verlopen en het verzamelen van ervaringsgegevens ten behoeve van toekomstige testprocessen.

Het eindrapport wordt na het geven van het vrijgaveadvies opgesteld, meestal wanneer het testobject al is vrijgegeven aan de volgende fase. Er is daarom meer tijd voor beschikbaar.

De inhoudsopgave van een eindrapport is min of meer gelijk aan een voortgangsrapport:

1. Evaluatie van het testobject (BDTM: resultaat)
   1.1. Status per kenmerk/deelobject
   1.2. Status testdoelen
2. Productrisico- en strategiebijstelling (BDTM: risico's)
3. Vrijgaveadvies (BDTM: resultaat, risico's)
4. Evaluatie van het testproces
   4.1. Voortgang (BDTM: tijd en kosten)
   4.2. Kwaliteit van het testproces (BDTM: alle aspecten)
6. Aanbevelingen voor toekomstige tests
7. Ervaringscijfers (optioneel)
8. Kosten/batenanalyse (optioneel)

Het eindrapport kijkt echter ten opzichte van een voortgangsrapport eerder terug dan vooruit. Ofwel, het gaat

voornamelijk in op het verschil tussen het oorspronkelijke plan en de uiteindelijke realisatie. In welke mate is afgeweken van het plan? Was het plan goed of zijn zaken verkeerd ingeschat? Is de bijsturing steeds tijdig en effectief geweest? In welke mate zijn de randvoorwaarden gedurende het testproces goed en tijdig ingevuld? Waren knelpunten te voorkomen geweest? Deze verschillen worden met name voor de risicoanalyse, teststrategie, begroting en planning geanalyseerd. Ook wordt naar de kwaliteit van het testproces gekeken: zijn de gekozen procedures, tools en technieken juist gebruikt en heeft de testomgeving voldaan aan de verwachting? Voor al deze punten worden zo mogelijk aanbevelingen gegeven voor toekomstige tests. De activiteit 'Evalueren testproces" (zie paragraaf 6.8.1) levert de input voor deze evaluatie. Ook kan gebruik gemaakt worden van de checklist "Evaluatie testproces". Daarnaast worden optioneel ervaringscijfers verzameld en ter beschikking gesteld van de opdrachtgever of, nog beter, een lijnorganisatie Testen. Een laatste optioneel onderdeel van het eindrapport is een kosten/batenanalyse.

Het eindrapport wordt beschikbaar gesteld aan de opdrachtgever en andere belanghebbenden en eventueel gepresenteerd.

Uitgediept

Ervaringscijfers

Voorbeelden van ervaringscijfers zijn:

- omvang van het testobject
- ontwikkelinspanning
- aantal bevindingen
- tijdsduur en uren per hoofdactiviteit
- tijdsduur en uren benodigd voor het specificeren van tests
- tijdsduur en uren benodigd voor het uitvoeren van tests
- aantal testgevallen
- analyse(doorloop)tijd per bevinding
- aantal te verwachten bevindingen
- aantal hertests.

Een uitgebreide opsomming van de mogelijk te verzamelen ervaringsgegevens is opgenomen in de lijst "Metricslijst" (zie hiervoor paragraaf 13.5). Daarnaast gaat dit hoofdstuk in op de Goal-Question-Metric methode voor het implementeren van metrics.

Kosten/batenanalyse

De kosten van het testproces zijn relatief eenvoudig vast te stellen. Denk hierbij aan de kosten van de gebruikte resources, mensen en middelen. De baten van het testproces zijn echter minder eenvoudig vast te stellen. Zoals in paragraaf 2.2 "Waarom testen" is aangegeven, zijn er vier soorten baten van het testen. Het is lastig, maar niet onmogelijk, om hier een meer cijfermatige uitspraak over te doen.

Producten

Rapportages (voortgangrapport, risicorapport, vrijgaveadvies, eindrapport).
Ervaringgegevens.
Kosten/batenanalyse.

Technieken

Checklist "Evaluatie testproces" (www.tmap.net).
Lijst "Metricslijst" (hoofdstuk 13).

Tools

Bevindingenbeheertool.
Testwarebeheertool.
Plannings- en voortgangsbewakingstool.

Workflowtool.

6.3.4 Bijsturen

Doel

Het, indien nodig na overleg met opdrachtgever, bijsturen van het testproces

Werkwijze

Wanneer de voorgestelde maatregelen zijn gerapporteerd en de opdrachtgever akkoord is en een keuze heeft gemaakt uit één of meer van de mogelijke alternatieven, kan de testmanager deze effectueren. Hiertoe voert hij de volgende stappen uit:

1. Doorvoeren maatregelen en evalueren effectiviteit
2. Aanpassen producten uit de planfase (optioneel, afhankelijk van tolerantie)
3. Terugkoppelen aan opdrachtgever

1. Doorvoeren maatregelen en evalueren effectiviteit

De testmanager voert in deze stap de gekozen (goedgekeurde) maatregelen door. Bovendien wordt na enige tijd geëvalueerd of met de genomen maatregelen het gewenste effect wordt bereikt.

2. Aanpassen producten uit de planfase (optioneel, afhankelijk van tolerantie)

De maatregelen kunnen gevolgen hebben voor de afspraken zoals die in het testplan zijn gemaakt. In dat geval past de testmanager de betreffende producten aan en legt de aangepaste producten ter goedkeuring aan de belanghebbenden voor.

Voorbeelden van wijzigingen in de verschillende producten zijn:

- De scope in de opdrachtformulering wordt aangepast. Dit is bijvoorbeeld het geval als besloten wordt om één of meer testvormen extra of juist niet uit te voeren.
- De productrisicoanalyse wordt herzien omdat tijdens de uitvoering van het testproces blijkt dat de foutkans verkeerd is ingeschat. Dit is bijvoorbeeld het geval als door tijdsdruk de ontwikkeltests slechts heel beperkt worden uitgevoerd.
- Tijdens de testuitvoering zullen in de teststrategie met name wijzigingen worden aangebracht als de testdiepgang of werkwijze aangepast wordt. Een voorbeeld hiervan: onder tijdsdruk wordt besloten om voor het testen van schermen geen testgevallen meer op te stellen, maar gebruik te maken van een checklist.
- Veel van de gebeurtenissen die genoemd zijn bij paragraaf 6.3.2 hebben gevolgen voor de begroting. Een veel voorkomend voorbeeld van een dergelijke gebeurtenis is het later opleveren van het testobject terwijl de deadline voor de test gelijk blijft. De geplande testdekking is dan slechts haalbaar door meer mensen in te zetten, met extra inwerkverliezen en management overhead tot gevolg.

Doordat de opdrachtformulering, productrisicoanalyse, teststrategie en begroting consistent met elkaar moeten zijn, zal een verandering in één van de producten veelal leiden tot veranderingen in de andere producten. Wijzigingen op het testplan worden vastgelegd in een nieuwe versie of in een aanvulling, dat opnieuw ter goedkeuring aan de opdrachtgever wordt voorgelegd. Het is de verantwoordelijkheid van de testmanager om de consequenties van de wijzigingen goed te communiceren met de opdrachtgever.

3. Terugkoppelen aan opdrachtgever

In deze stap rapporteert de testmanager over de genomen maatregelen en de gevolgen daarvan voor het testproces aan de

belanghebbenden, zoals bijvoorbeeld de opdrachtgever. Als de opdrachtgever (en eventueel andere belanghebbenden) al eerder betrokken zijn geweest bij het geven van toestemming om de maatregel te nemen, bevat deze rapportage doorgaans geen nieuwe informatie.

Ook als de testmanager de maatregel zelfstandig kan doorvoeren, wordt de gebeurtenis met bijbehorende maatregelen aan de belanghebbenden gerapporteerd om de belanghebbenden steeds inzicht te geven in het testtraject. De periodieke voortgangsrapportage is een geschikt middel voor deze rapportage.

Producten

Sturingsmaatregelen.
Bijgesteld plan.

Technieken

Niet van toepassing.

Tools

Plannings- en voortbewakingstool.
Workflowtool.

6.4 Fase Inrichting en beheer infrastructuur

Doel

Het beschrijven, realiseren en beheren van de testinfrastructuur die wordt gebruikt bij de verschillende fasen en activiteiten binnen TMap.

Context

De testinfrastructuur bestaat uit de faciliteiten en middelen die nodig zijn om de test adequaat te kunnen uitvoeren. Er wordt onderscheid gemaakt tussen de faciliteiten voor testuitvoering (testomgevingen), voor de ondersteuning van het testen (testtools) en voor het dagelijkse werk van de testers (werkplekken).

Definitie

De testinfrastructuur bestaat uit de faciliteiten en middelen die nodig zijn om de test adequaat te kunnen uitvoeren. Er wordt onderscheid gemaakt tussen testomgevingen, testtools en werkplekken.

Het inrichten en beheren van infrastructuur is een specifieke expertise. Het is iets waar testers over het algemeen beperkte kennis van hebben, maar waar ze wel heel erg van afhankelijk zijn (zonder infrastructuur geen test). Alle verantwoordelijkheden rond het inrichten en beheren van infrastructuur zijn daarom vaak bij een aparte beheerafdeling belegd. Bij een testtraject zal dus nauw met deze andere (soms ook externe) partijen moeten worden samengewerkt. Dit betekent dat testmanagers terecht komen in een situatie dat ze niet de bevoegdheden hebben rond de inrichting en beheer van de infrastructuur (dat ligt bij de beherende partij), maar daar wel van afhankelijk zijn. Dit kan tot conflicten leiden. Zo kan de situatie zich voordoen dat deze beherende partij voorrang geeft aan het oplossen van productieverstorende fouten boven het oplossen van fouten in een testomgeving. Daarnaast heeft een beheerafdeling vaak ook bepaalde security-richtlijnen (bijvoorbeeld autorisatiecontroles, vaste back-uptijden, installatieprocedures) waar niet zo maar van afgeweken kan worden. Dit is iets waarmee tijdens het testen rekening moet worden gehouden en daarmee is de zorg voor de inrichting en beheer van de infrastructuur een belangrijk aandachtsgebied voor de testmanager. Een oplossing, om de zorg voor dit ondersteunende proces te verlichten, is de permanente testorganisatie. Deze neemt de verantwoordelijk voor de inrichting en beheer van de testinfrastructuur voor zijn rekening. Dit wordt beschreven in hoofdstuk 8 “Ondersteunende processen”.

Voorbeeld

Bij een organisatie wordt de infrastructuur beheerd door een externe partij. Deze externe partij heeft als voorwaarde meegekregen dat van de infrastructuur dagelijks een back-up gemaakt moet worden. Hiervoor wordt een geautomatiseerd proces opgesteld dat ergens in de nachtelijke uren (tussen 22.00 – 06.00), afhankelijk van andere processen, een back-up maakt.

Bij de bouw en test van een nieuwe webapplicatie ontstaat uitloop en er wordt voor gekozen om per dag langer te gaan testen. Dit betekent dat de testers (in ploegendienst) van 06.00 tot 01.00 gaan testen. Het is daarom noodzakelijk om de tijden van het back-up proces aan te passen. Hiervoor wordt een verzoek ingediend, maar de externe organisatie kan het verzoek niet zomaar inwilligen. Er moeten veel andere processen aangepast worden en dat kan wel 2 weken in beslag nemen. De optie om van de testomgeving geen back-up te maken is onbespreekbaar om allerlei juridische redenen. Ondertussen staat de testmanager onder druk om een oplossing te vinden voor het probleem van de uitloop.

Bij een testproject is het belangrijk speciale aandacht te schenken aan het inrichten en beheren van de infrastructuur. Om daar de focus gedurende de test op te houden is het een aparte fase binnen het faseringsmodel van TMap. Het is een fase die parallel loopt aan de fasen Voorbereiding, Specificatie, Uitvoering en Afronding. Voor sommige activiteiten bestaan afhankelijkheden met activiteiten in de andere TMap-fasen. Dit wordt later in deze paragraaf bij deze betreffende activiteit zelf toegelicht.

Testomgeving

Voor het dynamisch testen van een testobject (runnen van software) is een passende testomgeving nodig.

Definitie

Een testomgeving is een samenstelling van onderdelen zoals hard- en software, koppelingen, omgevingsdata, beheertools en processen waarin een test wordt uitgevoerd.

Onder hardware worden alle tastbare onderdelen van een computer verstaan (scherm, harde schijf, netwerkkaart, enzovoort). Testomgevingssoftware zijn alle programma's die op de beschikbare hardware aanwezig moeten zijn om de te testen software te kunnen runnen, zoals besturingssoftware, DBMS, netwerk en andere hulpprogramma's. Koppelingen omvatten alles wat nodig is om het testobject met andere systemen te laten communiceren. De omgevingsdata is de set aan gegevens die de testomgeving nodig heeft om ermee te kunnen werken (gebruikersprofielen, netwerkadressen, stamtabellen enzovoort). Beheertools zijn tools die specifiek nodig zijn om de testomgeving operationeel te houden. En processen omvatten alle activiteiten die uitgevoerd worden rond het inrichten en beheren van een testomgeving.

In paragraaf 8.4 "Testomgevingen" wordt dieper ingegaan op testomgevingen.

Testtools

Definitie

Een testtool is een geautomatiseerd hulpmiddel dat ondersteuning biedt aan één of meer testactiviteiten, zoals plannen, beheren, specificeren en uitvoeren.

Testtools kunnen als instrument worden gebruikt om hogere productiviteit en/of effectiviteit van de testers en de test te bereiken. De nadruk bij het gebruik van testtools ligt op "ondersteunen" (zie ook de definitie). Dit betekent dat een testtool pas een hulpmiddel is als het gebruik ervan iets oplevert; het mag geen doel op zich zijn om een tool te gebruiken.

Eén van de voorwaarden voor succesvol gebruik van testtools is de aanwezigheid van een gestructureerde testaanpak. In een goed beheerst proces kunnen tools zeker een belangrijke meerwaarde geven, maar ze werken contraproductief bij een onvoldoende beheerst testproces. De reden hiervoor is dat automatisering (wat in feite testtools doen) vraagt om een

zekere herhaalbaarheid en standaardisatie van de te ondersteunen activiteiten. Een ongestructureerd proces kan niet aan deze voorwaarden voldoen. De inzet van testtools kan wel als hefboom fungeren om een gestructureerde aanpak te implementeren. Structurering en automatisering moeten dan echter minimaal hand in hand gaan, kortom: “Structure and Tool”.

In paragraaf 8.5 “Testtools” wordt dieper ingegaan op testtools.

Werkplekken

Een van de aspecten die bij het testen vaak wordt vergeten, is het beschikbaar stellen van een werkplek, waar testers hun werk onder goede omstandigheden effectief en efficiënt kunnen uitvoeren. Het gaat hierbij om kantoorinrichting in de breedste zin van het woord, want ook de testers moeten hun werk onder goede omstandigheden kunnen uitvoeren. De werkplek omvat dus meer dan alleen kantoorruimte en een PC. Ook zaken als bijvoorbeeld toegangspasjes, stroomvoorziening en faciliteiten voor het nuttigen van de lunch moeten geregeld worden.

Wanneer het testen op projectmatige basis wordt uitgevoerd dan moet er extra kantoorruimte worden geregeld. Het kan raadzaam zijn het testteam in één locatie (kamer, verdieping) onder te brengen. Dit vormt dan de basis voor een goede onderlinge samenwerking en afstemming binnen het team. Wanneer dat niet mogelijk is, dan dient de verdeling van de teamleden over verschillende kamers te zijn afgestemd op bijvoorbeeld de verdeling van de testers over de verschillende systeemdelen, de toe te passen testvormen en dergelijke. Wanneer ontwikkelaar en tester samenwerken in multidisciplinaire teams, dan moeten deze in één locatie worden ondergebracht.

Zoals bij elke projectactiviteit wordt ook bij het testen veelvuldig overleg gepleegd. Doordat testen zich begeeft op het kruispunt van de verschillende activiteiten in het project hebben testers veel contact met de verschillende groepen (zoals ontwerpers, programmeurs, beheerders en gebruikers). Het is raadzaam het testteam in de nabijheid van deze verschillende groepen te plaatsen. Er zijn voorbeelden van een beter testproces door de verhuizing van het testteam naar het fysieke ‘midden’ van de projectorganisatie. Hierdoor werd onder andere het wederzijds respect tussen de testers en andere projectdeelnemers vergroot wat de kwaliteit altijd ten goede kwam.

De werkplek voor een tester wijkt zo op het eerste gezicht niet veel af van de reguliere werkplek. Maar schijn bedriegt. Hetgeen getest wordt is vaak nieuw voor de organisatie en de werkplek. Testers kunnen dan te maken krijgen met de situatie dat hun werkplek nog niet is voorbereid op de nieuwe software. Daarom is het vaak nodig om voor testers aparte autorisaties te regelen. Zo moeten testers bijvoorbeeld de mogelijkheid krijgen om de nieuwe software te installeren op hun lokale PC. Maar dit kan ook noodzakelijk zijn om bepaalde testtools te gebruiken.

Tip

Bepaalde testvormen kunnen enorm veel gegevens opleveren. Voorbeeld hiervan is de performancetest waarbij gebruik wordt gemaakt van een testtool. De output van deze testtool kan bestaat uit duizenden regels informatie. Opgeslagen in bestanden kan dit wel oplopen tot meerdere gigabytes per test. Uitgeprint zijn het vaak boekwerken van wel meer dan honderd pagina’s dik. Het is daarom aan te raden om hier aparte maatregelen voor te nemen. Zo kan er extra schijfruimte worden gereserveerd en een extra printer worden aangesloten.

Randvoorwaarden

Voordat er gestart kan worden met de fase Inrichting en beheer infrastructuur dient de beschrijving van de benodigde infrastructuur op overkoepelend niveau bekend te zijn, inclusief globale planning, vastgelegd in het testplan (zie paragraaf 6.2.11 “Definiëren infrastructuur”) en/of mastertestplan (zie paragraaf 5.2.9 “Definiëren infrastructuur”). Wanneer gebruik wordt gemaakt van testtools dient bekend te zijn hoe de verschillende activiteiten binnen TMap ingevuld worden.

Werkwijze

Aan de hand van de in het testplan opgenomen definitie van de infrastructuur wordt bekeken of een nadere specificatie en detaillering noodzakelijk is. Naast het beschrijven van de benodigde middelen, wordt ook beschreven wat verwacht wordt van de leverende partijen tijdens het beheer van deze middelen. Doordat er andere expertise voor nodig is, wordt het realiseren van de infrastructuur vaak door andere partijen uitgevoerd. Vanuit het testproject wordt de voortgang van de

realisatie wel gemonitord. Indien deze voortgang in gevaar komt worden acties uitgezet. De realisatie moet gereed zijn voordat de fase Uitvoering start maar bij voorkeur eerder. Tegelijk met de realisatie van de infrastructuur wordt een checklist opgesteld met daarin specifieke controles. Hiermee wordt bij oplevering bepaald of de geleverde infrastructuur voldoet aan de eerder gestelde eisen. Na oplevering moet de infrastructuur beschikbaar gehouden worden voor de testers tegen het kwaliteitsniveau dat bij aanvang van de fase is bepaald. Bij beëindiging van de testopdracht wordt onderzocht welke onderdelen van de infrastructuur geconserveerd moet worden. Deze kunnen dan bij toekomstige (her)tests weer gebruikt worden.

Rollen / verantwoordelijkheden

Een aanbeveling is om de verantwoordelijkheid voor de invulling van deze fase te delegeren aan iemand anders door de testmanager neer te leggen. Deze persoon heeft dan de rol van testinfrastructuurcoördinator. Deze rol wordt in hoofdstuk 16 “Testrollen” toegelicht.

Activiteiten

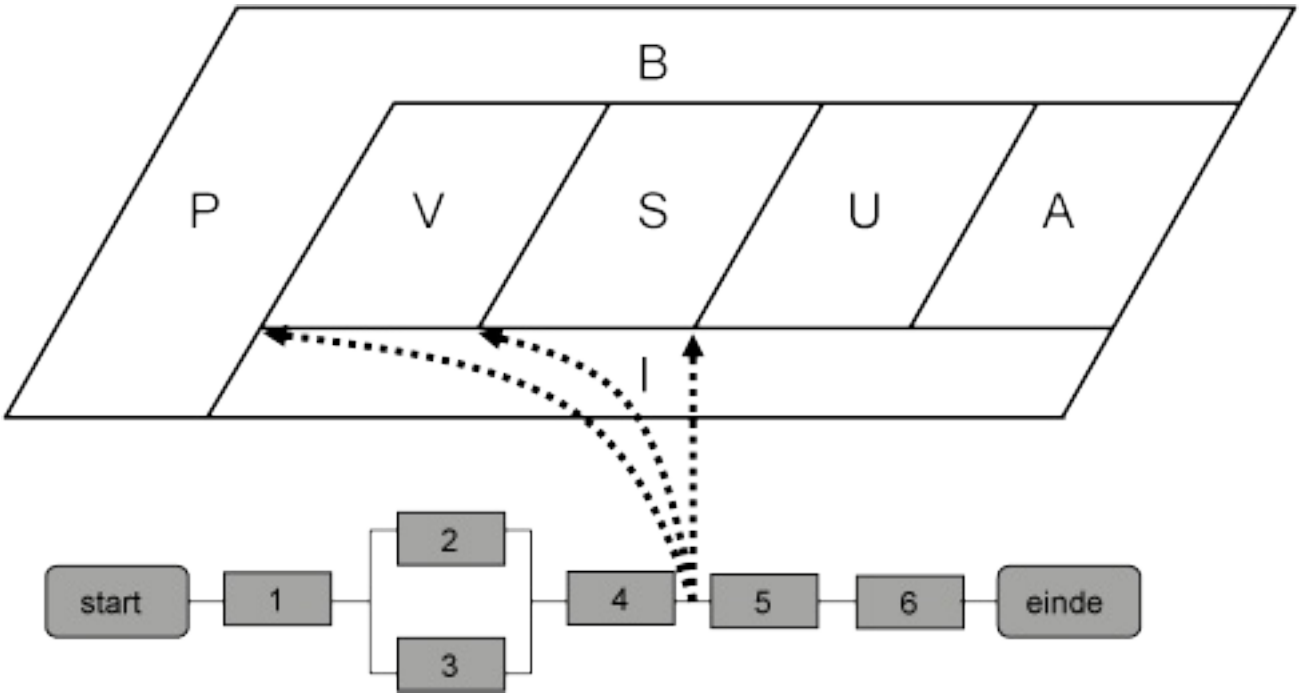
De basis voor de fase Inrichting en beheer infrastructuur wordt gelegd in de fase Planning. Hier wordt in de activiteit “Definiëren infrastructuur” een beschrijving gemaakt van de benodigde infrastructuur op overkoepelend niveau, inclusief planning. Deze beschrijving (uit het mastertestplan of testplan) dient als input voor de eerste activiteit in deze fase.

De fase Inrichting en beheer infrastructuur bestaat uit de volgende 6 activiteiten:

1. Specificeren infrastructuur;
2. Realiseren infrastructuur;
3. Specificeren intake infrastructuur;
4. Intake infrastructuur;
5. Beheren infrastructuur;
6. Conserveren infrastructuur.

Figuur 6.15 “Inrichting en beheer infrastructuur” geeft de volgorde en de afhankelijkheid aan tussen de verschillende activiteiten. Activiteit “Realiseren infrastructuur” en activiteit “Specificeren intake infrastructuur” kunnen parallel worden uitgevoerd. Belangrijk is de afhankelijkheid tussen het einde van activiteit “Intake infrastructuur” en de start van de fase Uitvoering. Voordat er gestart kan worden met de testuitvoering moet er een correct werkende testinfrastructuur zijn.

Daarom is het van groot belang activiteit “Intake infrastructuur” te plannen voor de start van de testuitvoering. Aan te raden is om dit zelfs ver daarvoor te plannen (en dus de voorliggende activiteiten ook) om eventuele opstartproblemen van de testinfrastructuur niet meteen door te laten werken in uitloop van de testuitvoering. Testuitvoering ligt vaak op het kritieke pad van het gehele project en dus zorgen problemen met de testinfrastructuur indirect ook voor uitloop van het project. Bovendien is een operationele infrastructuur erg handig bij de fase Specificatie. Testscripts kunnen al uitgeprobeerd worden en testgegevens (in bijvoorbeeld bestanden) ingevoerd.



Figuur 6.15: Inrichting en beheer infrastructuur.

In de definitie van testinfrastructuur staat dat onderscheid wordt gemaakt in testomgeving, testtool en werkplek. Voor ieder van deze drie moet het model van activiteiten, zoals het hiervoor staat beschreven worden ingevuld. Onderling hebben de activiteiten van de drie onderdelen een relatie met elkaar voor wat betreft tijdigheid. Hier speelt activiteit “intake infrastructuur” een belangrijke rol. De intake van de infrastructuur vormt de koppeling met de andere fasen binnen TMap en is tegelijk de koppeling van de drie onderdelen onderling.

Het is aan te raden de werkplek zo snel mogelijk te regelen en deze moet al aanwezig zijn voordat de testers komen. Dit moet dus al geregeld worden tijdens de fase Planning. Vaak is het zelfs zo dat de werkplek operationeel moet zijn voordat begonnen kan worden met de intake van de testomgeving. En zo moet de testomgeving op zijn beurt vaak operationeel zijn voordat begonnen kan worden met de intake van de testtool. Dit wordt in figuur 6.15 “Inrichting en beheer infrastructuur” duidelijk gemaakt. Het inrichten en beheren van de infrastructuur is een zeer complexe zaak met veel interne en externe afhankelijkheden. Het organiseren vraagt om veel aandacht en het is daarom aan te raden dit bij de testinfrastructuurcoördinator te beleggen.

Tip

Wanneer testtools voor geautomatiseerde testuitvoering (zie ook paragraaf 8.5.3 “Soorten testtools”) worden gebruikt dan moet de testinfrastructuur operationeel zijn voordat de fase Specificatie start. Dit betekent concreet dat de activiteit “intake infrastructuur” afgerond moet zijn voordat de fase Specificatie start. De reden hiervoor is dat tijdens de fase Specificatie de geautomatiseerde testscripts geprogrammeerd worden en hiervoor is een operationele werkplek, testomgeving en testtool nodig.

6.4.1 Specificeren infrastructuur

Doel

Het verder detailleren van de beschrijving van de benodigde infrastructuur met bijbehorende realisatieplanning en beheerafspraken.

Werkwijze algemeen

De beschrijving van de benodigde infrastructuur op (master)testplan niveau wordt in deze activiteit verder uitgewerkt. Ook wordt voor de testomgeving, testtools en werkplek de planning verder gedetailleerd. Naast het beschrijven van wat er nodig is in middelen wordt ook beschreven wat verwacht wordt van de leverende partijen tijdens het beheer van deze middelen. Aan de hand van de in het testplan opgenomen beschrijving van de infrastructuur wordt bekeken of een nadere specificatie en detaillering noodzakelijk zijn. Het tijdig betrekken van de verschillende partijen is essentieel. Er moeten afspraken gemaakt worden voor het leveren en opbouwen van de infrastructuur en deze afspraken moeten steeds tussentijds worden gecontroleerd. In overleg met de verschillende leveranciers (intern en extern) wordt bepaald hoe gedetailleerd de specificatie moet zijn. De levertijden van de diverse onderdelen worden in de detailplanning opgenomen.

Werkwijze werkplek

Het specificeren van de werkplek omvat tastbare onderwerpen als benodigde ruimtes, bureaus, stoelen, telefoons, PC's enzovoort. Maar ook de minder tastbare zaken als benodigde autorisaties, schijfruimte, software, e-mailaccounts enzovoort. Het realiseren van deze zaken kan een lange doorlooptijd vergen. Soms moet het speciaal opgezet worden (bijvoorbeeld projectruimtes) of speciaal geïnstalleerd worden (bijvoorbeeld de PC's). In andere gevallen moeten zaken gewoon nog besteld worden. Het is aan te raden om hier al specifieke eisen en wensen te benadrukken die worden gesteld aan het beheer van de werkplek. Hierbij kan gedacht worden aan het verkrijgen van een aparte status bij het oplossen van problemen op de werkplek voor de testers. Dit kan handig zijn omdat testers nu eenmaal geen 'gewone gebruikers' zijn en dus soms andere soort ondersteuning nodig hebben .

Werkwijze testomgeving

Bij het specificeren van de testomgeving moet gekeken worden naar de verschillende componenten van de testomgeving. Definities kunnen per leverancier en organisatie verschillen. Daarom moet altijd goed besproken worden wat onder een bepaalde term verstaan wordt. Een ander belangrijk punt is het aantal testomgevingen dat nodig is en de verschillende soorten daarin. Iedere soort testomgeving heeft zijn eigen doel waarvoor dus specifieke eisen gelden. Het specificeren van de technische invulling van de testomgeving moet in overleg met iemand die de technische kennis heeft van de omgevingen. Deze persoon moet de concrete eisen en wensen (gebaseerd op het doel van de test waarvoor de testomgeving dient) vertalen naar de technische invulling. Als basis hiervoor kan bijvoorbeeld een architectuurplaat worden opgesteld.

Dit kan een moeizaam proces zijn omdat twee werelden (test en techniek) twee verschillende talen spreken. Het is aan de verantwoordelijke vanuit het testteam (de testmanager of testinfrastructuurcoördinator) de taak om te controleren of hieraan een goede invulling wordt gegeven.

Naast eisen over de opzet van de testomgeving moeten ook eisen worden opgesteld voor het beheer. Voorbeelden van eisen zijn:

- de back-up activiteiten die uitgevoerd moeten worden;
- de inzichtelijkheid van de aanwezige versies van programmatuur;
- de aanwezige interfaces;
- het kunnen wijzigen van de testomgeving;
- het kunnen wijzigen van de systeemdatum;
- het gebruik van testgegevens en het beheer ervan;
- autorisaties en het beheer daarvan;
- de benodigde tijdsduur voor het opbouwen van een testomgeving.

Ook moeten er nu al afspraken worden gemaakt over hoe de testomgeving getest gaat worden (zie ook activiteit intake infrastructuur). Andere afspraken kunnen gaan over het contact met leveranciers (rechtstreeks door testteam of via andere partij) en hoe om te gaan met licenties.

Voorbeeld

Bij een test van een nieuwe klantenadministratie worden tijdens de specificatie de volgende eisen gesteld aan de testomgeving en het beheer ervan:

- Back-ups worden gemaakt op aanvraag van de testers en duren niet langer dan 15 minuten;
- Er worden geen wijzigingen in de omgeving doorgevoerd zonder expliciete toestemming via mail van de testcoördinator;
- Gemaakte back-ups worden op aanvraag van de testcoördinator teruggezet binnen 15 minuten;
- Het terugzetten en veiligstellen van omgevingen gebeurt tussen 20.00 – 06.00;
- Het operatingsysteem op de testomgeving is hetzelfde als op de productieomgeving;
- Koppeling met de testomgeving van systeem X en Y moet tussen 06.00 – 20.00 beschikbaar zijn;
- Koppeling met de testomgeving van systeem Z moet tussen 06.00 – 20.00 op afroep binnen 15 minuten beschikbaar zijn;
- Koppeling met systeem W wordt gesimuleerd door een stub;
- Testers hebben rechtstreeks toegang tot tabellen in de database (leesrechten);
- De systeemdatum moet door het testteam gewijzigd kunnen worden;
- Er moeten 4 versies van testbestanden bewaard kunnen worden;
- De tool A moet beschikbaar zijn voor het creëren of kopiëren van complete testgevallen.

Tip

In sommige organisaties wordt gebruik gemaakt van een standaard set aan testomgevingen. De testmanager moet hier dan voor zijn test gebruik van maken. Wanneer dat het geval is, wordt tijdens deze activiteit onderzocht wat de specifieke kenmerken zijn van deze testomgevingen en hoe deze binnen het testtraject passen.

Werkwijze testtool

Wanneer bij het opstellen van het (master)testplan de beslissing is genomen om gebruik te maken van testtools, dan dient dit in deze activiteit verder geconcretiseerd te worden. Er moet invulling gegeven worden aan deze beslissing en er moeten concrete keuzes gemaakt worden voor één of meer tools. Zoals ook uit definitie van testtools naar voren komt dienen deze ter ondersteuning van één of meer testactiviteiten. Tijdens deze specificatie van de testtools moet precies duidelijk zijn welke testactiviteiten ondersteund dienen te worden en hoe dit er uit dient te zien.

Het selecteren en inzetten van een tool kan volgens het faseringsmodel wat staat beschreven in paragraaf 8.5.5 “Invoeren van testtools met toolbeleid”.

Producten

Detailspecificatie werkplek.
Detailspecificatie testomgeving.
Plan van aanpak testtool(s).

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.4.2 Realiseren infrastructuur

Doel

Het realiseren van de infrastructuur op basis van de specificatie die is opgesteld in de vorige activiteit.

Werkwijze

De gedetailleerde specificatie uit de vorige activiteit wordt in deze activiteit gerealiseerd. De benodigde hardware en software worden indien nodig aangeschaft of besteld. De werkplekken worden gerealiseerd, de testomgeving ingericht en de testtools geïnstalleerd en geconfigureerd. Wanneer gebruik wordt gemaakt van testtools voor geautomatiseerde testuitvoering dan wordt tijdens deze activiteit het framework van de testsuite gerealiseerd (zie paragraaf 8.5.3 “Soorten testtools” voor meer informatie over een framework van de testsuite). Omdat dit allemaal speciale expertise vereist, wordt dit meestal uitgevoerd door andere partijen dan de testers. Vanuit het testproject (de testinfrastructuurcoördinator) moet de voortgang van de realisatie wel gemonitord worden indien deze voortgang in gevaar komt.

Deze activiteit moet parallel worden uitgevoerd met de eerste fasen van TMap en moet uiterlijk gereed zijn voordat de fase Uitvoering begint (bij voorkeur nog daarvoor, want er is tijd nodig voor de volgende activiteit “intake infrastructuur”). Wanneer de activiteit wordt uitgevoerd, is afhankelijk van het onderdeel dat wordt gerealiseerd en van de afhankelijkheden tussen de verschillende onderdelen. Zo dient de werkplek als eerste te worden gerealiseerd, bij voorkeur in de fase Planning. Het realiseren van de testomgeving kost vaak veel tijd en moet daarom snel gestart worden. Maar de situatie kan zich voordoen dat voor het realiseren van de testomgeving een werkplek nodig is. In dat geval moet dus gewacht worden tot de realisatie van de werkplek is voltooid. Wanneer de testtool gebruik maakt van de testomgeving kan de realisatie (installatie en configuratie) pas starten wanneer de testomgeving is gerealiseerd. Anders is het ook aan te raden hier zo snel mogelijk mee te beginnen.

Interne maar ook externe partijen (bijvoorbeeld de leverancier van de testtool) spelen hier een rol. Dit maakt het een moeilijk beheersbare activiteit, dat vraagt om een goede coördinatie. De infrastructuurcoördinator moet controleren op de voortgang en de kwaliteit van het geleverde werk. Zo moeten de volgende deelactiviteiten worden uitgevoerd:

- het controleren of alle afspraken nog gelden;
- het laten oplossen van knelpunten, problemen en vastleggen van genomen maatregelen in nieuwe afspraken;
- het checken van installaties. Hiervoor kan (voor zover mogelijk) al gebruik worden gemaakt van de opgestelde checklists ten behoeve van de intake infrastructuur (zie activiteit “specificeren intake infrastructuur”, paragraaf 6.4.3).

Producten

Operationele werkplek.
Operationele testomgeving.
Geïnstalleerde testtools.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.4.3 Specificeren intake infrastructuur

Doel

Het vastleggen van de wijze waarop de intake van de infrastructuur wordt uitgevoerd.

Werkwijze

Doordat de infrastructuur vaak door andere partijen dan het testteam wordt geleverd en het een zeer belangrijke rol speelt binnen het verdere traject, is het van belang een formeel acceptatiemoment te benoemen. Bij dit formele acceptatiemoment wordt bepaald of de producten functioneren voor verder gebruik en of ze voldoen aan de eerder gestelde eisen en wensen (het is een soort acceptatietest van de infrastructuur). Deze acceptatie gebeurt door middel van een intake; een activiteit, waar op basis van een checklist met controles, wordt bepaald of de werkplek, de testomgeving of de testtool functioneert en of het voldoet aan de eerder gestelde eisen en wensen.

De checklist wordt opgesteld, gebaseerd op de specificaties van de verschillende onderdelen. De checklist moet uiterlijk beschikbaar zijn zodra de vorige activiteit (realiseren infrastructuur) van het betreffende onderdeel beëindigd is, maar bij voorkeur eerder. Zo kan de checklist al tijdens de realisatie gebruikt worden voor tussentijdse controles.

De activiteit heeft een nauwe relatie met de activiteit “Specificeren intake testobject” in de fase Specificatie. Er zijn situaties waar bepaalde aspecten van een infrastructuur slechts gecontroleerd kunnen worden met behulp van (een vroege of tussenversie van) het testobject. Zo kan bijvoorbeeld een releaseprocedure van de testomgeving alleen gecontroleerd worden met het testobject. Maar ook de correcte installatie van een testtool voor de automatisering van de uitvoering kan alleen gecontroleerd worden met het testobject.

Voorbeeld

De volgende controles kunnen worden uitgevoerd van de werkplek:

- Zijn de benodigde pc’s, printers, werkplekken, telefoonlijnen, routers e.d. aanwezig en correct geïnstalleerd?
- Is de benodigde systeemsoftware geïnstalleerd?
- Heeft de systeemsoftware de juiste versie?

De volgende controles kunnen worden uitgevoerd van de testomgeving:

- Is toegang tot de testomgeving geregeld?
- Is toegang tot de applicatie geregeld?
- Is toegang tot de database geregeld?
- Is de database gevuld met de juiste data (bijvoorbeeld kopie productie)?
- Zijn alle autorisaties geregeld?

De volgende controles kunnen worden uitgevoerd van de testtools:

- Zijn alle licenties operationeel?
- Kan vanaf elke werkplek de testtool worden benaderd?
- Is de connectie tussen testtool en testobject operationeel?

Producten

Checklist “intake werkplek”.
Checklist “intake testomgeving”.
Checklist “intake testtools”.
Procedure intake.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool.

6.4.4 Intake infrastructuur

Doel

Het uitvoeren van de intake zoals die in de vorige activiteit is voorbereid.

Werkwijze

Alle controles op de checklist, die in de vorige activiteit zijn opgesteld, worden afgelopen. Hiermee wordt bepaald of de testomgeving, testtool of werkplek functioneren en of het ze voldoen aan de eerder gestelde eisen en wensen. Eventuele ontbrekende delen worden door middel van een bevinding (zie hoofdstuk 12 “Bevindingenbeheer”) gerapporteerd aan de betrokken partijen. Uiteraard moeten deze delen dan zo spoedig mogelijk beschikbaar gesteld worden. Ontbrekende delen voor de testomgeving zullen vertragend werken en impact hebben op het gehele project. De fase Uitvoering ligt immers vaak op het kritieke pad en wanneer die niet kan starten (doordat bijvoorbeeld de testomgeving niet functioneert) loopt het gehele project vertraging op. De intake mag niet onderschat worden en moet zo snel mogelijk worden uitgevoerd. De intake van de testomgeving wordt bij voorkeur tijdens de fase Specificatie uitgevoerd. Wanneer dit niet mogelijk is, dan uiterlijk bij aanvang van de fase Uitvoering. Dit kan het geval zijn wanneer het testobject nodig is en deze op dat moment pas beschikbaar is.

Producten

Testbevindingen;
Operationele en bruikbare werkplek (Checklist “intake werkplek”);
Operationele en bruikbare testomgeving (Checklist “intake testomgeving”);
Operationele en bruikbare testtool (Checklist “intake testtools”);
Intakeverslag.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool;
Bevindingenbeheertool.

6.4.5 Beheren infrastructuur

Doel

Het beschikbaar houden van de infrastructuur (testomgeving, testtools en werkplekken) voor de testers.

Werkwijze werkplek

Het beheren van de werkplek, zodat deze beschikbaar is en blijft voor de testers, is meestal een activiteit die standaard wordt geregeld in andere beheeractiviteiten binnen een organisatie. Voor wat betreft de PC op de werkplek is het belangrijk dat de standaard beherende organisaties weten dat deze speciaal voor testers is. Want dit kan betekenen dat er andere afspraken gelden rond bijvoorbeeld autorisaties en prioriteitstelling voor het oplossen van problemen.

Werkwijze testtool

De testtool kan beheerd worden binnen het testproject door de testers die er gebruik van maken, maar ook door een aparte beheerafdeling (bijvoorbeeld een permanente testorganisatie). Een belangrijke beheercomponent is het regelmatig controleren op nieuwe versies van de testtool en deze dan beschikbaar stellen aan de gebruikers. Daarnaast gelden de beheeractiviteiten zoals ze bij de testomgeving staan beschreven ook voor de testtool.

Werkwijze testomgeving

Het beschikbaar stellen en houden van de testomgeving zodat de testers hun testgevallen kunnen uitvoeren en bevindingen analyseren omvat een scala aan activiteiten. Deze vinden plaats tijdens de fase Uitvoering. Voorbeeld hiervan zijn:

- Oplossen knelpunten;
- Beschikbaar stellen logging;
- Back-up & restore;
- Doorvoeren wijzigingen;
- Monitoren.

Meer informatie over het beheer van testomgevingen en de bijbehorende processen staat beschreven in paragraaf 8.4 “Testomgevingen”.

Oplossen knelpunten

De uitvoering van testscripts kan vertraging oplopen doordat er problemen optreden in de testomgeving (bijvoorbeeld: een batchprogramma heeft niet gedraaid). Doordat de uitvoering van testscripts vaak op het kritieke pad liggen van een project is het zaak deze knelpunten met de hoogste prioriteit op te lossen.

Voorbeeld

Bij een overheidsinstelling loopt een project met een vaste deadline omdat de oplossing gerelateerd is aan een wetswijziging die voor een bepaalde datum geïmplementeerd moet zijn. De fase Uitvoering ligt op het kritieke pad van dit project en het is daarom in ieders belang dat deze fase geen vertraging oploopt. In overleg met de beheerafdeling wordt daarom afgesproken de infrastructuur waar de testers gebruik van maken een zogenaamde “productie”-status te geven. Dit houdt in dat de beheerafdeling knelpunten die testers aandragen met dezelfde prioriteit afhandelt alsof het knelpunten zijn in de productieomgeving. Dit wordt gerechtvaardigd door het feit dat wanneer het project niet op tijd klaar, de wetswijziging niet op tijd geïmplementeerd is en dus het primaire proces geen doorgang meer kan vinden. Deze status aparte geldt alleen tijdens de testuitvoering.

Beschikbaar stellen logging

Systemen kunnen informatie vastleggen in de vorm van logging. Deze logging kan achteraf gebruikt worden ter controle van de acties die zijn uitgevoerd. De logging is een belangrijke informatiebron voor testers bij de analyse van hun bevindingen. Het beschikbaar stellen van deze informatie is daarom ook een belangrijke activiteit. Er kan voor gekozen

worden deze op afroep beschikbaar te stellen, maar een andere (en minder arbeidsintensieve) variant is de testers zelf toegang te geven tot de logging.

Back-up & restore

Juist van een infrastructuur die door testers wordt gebruikt is het belangrijk regelmatig de data veilig te stellen door middel van back-ups. Dit kan om uitgangssituaties veilig te stellen en deze keer op keer weer te gebruiken voor de test, maar ook om bepaalde bevindingen te onderzoeken. Het handelt hier niet alleen om back-ups van de testomgeving, maar ook om die van testtools en de PC's op de werkplek.

> **Tip**
>
> Voer altijd een test uit van de back-up & restore voordat de testuitvoering start. Dit kan door de eerste keer dat een back-up wordt gemaakt deze meteen te restoren.

Doorvoeren wijzigingen

Tijdens het project is de testomgeving door allerlei interne en externe oorzaken onderhevig aan veranderingen door bijvoorbeeld:

- gefaseerde oplevering en wijzigingen in de testomgeving;
- oplevering of heroplevering van (delen van) het testobject;
- nieuwe of veranderde procedures;
- wijzigingen in de simulatie- en systeemprogrammatuur;
- wijzigingen in de apparatuur, protocollen, parameters, enzovoort;
- nieuwe of gewijzigde testtools;
- wijzigingen in de testbestanden, tabellen, enzovoort:
  - converteren van testinvoerbestanden naar een nieuw formaat;
  - reorganisatie van testbestanden;
  - veranderingen in naamgeving.

Wijzigingen in de testomgeving mogen alleen worden doorgevoerd na toestemming van het testmanagement. Afhankelijk van de aard en omvang van de wijziging zal dit algemeen bekend worden gemaakt aan het testteam. Er vindt dan een nieuwe intake plaats van de testomgeving.

> **Tip**
>
> Een planningsvalkuil is aan te nemen dat het installeren van een nieuwe versie van het testobject geen tijd kost. In een bepaald project namen de eerste paar versies steeds weken in beslag door de enorme complexiteit én instabiliteit van het geheel van testomgeving en testobject. Later werd dit geoptimaliseerd tot telkens enkele dagen.

Monitoren

Soms kan de situatie ontstaan dat een bevinding meer onderzoek vergt en dat daar meer diepgaande technische kennis voor nodig is dan waar de tester over beschikt. Hiervoor kan de hulp worden ingeroepen van de beheerder die kan 'meekijken' op technisch niveau (monitoren) wat er gebeurt bij bepaalde handelingen.

Producten

Operationele en beheerde testinfrastructuur.
Bevindingen testinfrastructuur.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.4.6 Conserveren infrastructuur

Doel

Het identificeren, actualiseren en overdragen van de te beheren infrastructuur, zodanig dat bij toekomstige (her)tests hier weer gebruik van gemaakt kan worden.

Werkwijze

Deze activiteit is optioneel. Wanneer de activiteit wordt uitgevoerd start deze tegelijk met de fase Afronding. De activiteit omvat de volgende deelactiviteiten:

1. Selecteren infrastructuur;
2. Verzamelen en bijwerken infrastructuur;
3. Overdragen infrastructuur.

1. Selecteren infrastructuur

In overleg met de toekomstige beheerder van de infrastructuur wordt geïnventariseerd welke onderdelen nu werkelijk gebruikt zijn (de configuratie) en welke 'waard zijn' over te dragen. De afweging moet gemaakt worden tussen, wat het kost om de infrastructuur te bewaren en beheren, en wat het kost om de infrastructuur later weer te realiseren. Daarnaast bestaat de mogelijkheid dat bepaalde software of hardware (zoals onderdelen van testomgeving, maar ook bepaalde testtools) alleen maar in de beginfase van het testtraject gebruikt zijn en nu niet meer nodig zijn. Het is dan verspilde moeite om deze in beheer te nemen. Door deze identificatie kan ook duidelijk worden wat het verschil nu is tussen de gespecificeerde infrastructuur en de werkelijk gebruikte infrastructuur. Hier kan verschil in zitten (bepaalde software of hardware die wel is ingericht maar nooit is gebruikt) en dit leerpunt kan meegenomen worden in de evaluatie van het testproces (zie activiteit "evalueren testproces" in de fase Afronding, paragraaf 6.8.1).

2. Verzamelen en bijwerken infrastructuur

De beschrijving van de infrastructuur in de "Detailspecificatie infrastructuur" moet worden aangepast aan de configuratie die overgedragen moet worden. Dit is van essentieel belang omdat anders bij toekomstige testtrajecten alles opnieuw bedacht moet worden. Het is belangrijk om bij deze beschrijving ook goed te kijken naar de configuratie van de werkplekken. In deze "Detailspecificatie infrastructuur" wordt een (pak)lijst opgenomen met daarin de componenten die overgedragen worden. Componenten kunnen zijn licenties, omgevingsdata, scripts, software, tools, registry-files, hardware, accounts, databases, files enzovoort.

3. Overdragen infrastructuur

Tenslotte vindt de werkelijke overdracht van de infrastructuur plaats. Conform de aangepaste lijst in het document "Detailspecificatie infrastructuur" wordt de configuratie overgedragen.

Producten

Geconserveerde testinfrastructuur.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.5 Fase Voorbereiding

Doel

Het kunnen beschikken over een, met de opdrachtgever van de test overeengekomen, testbasis die voldoende van kwaliteit is voor het ontwerpen van de testgevallen. Voor het bepalen hiervan wordt in deze fase de detailintake van de testbasis uitgevoerd. Deze fase geeft inzicht in de testbaarheid van het systeem.

Definitie

Testbaarheid is het gemak en de snelheid waarmee kenmerken van het systeem (na iedere aanpassing) kunnen worden getest.

Vroege foutdetectie

Naast vaststellen van testbaarheid van de testbasis is er nog een reden om de testbasis te beoordelen (toetsen). Met toetsactiviteiten kunnen namelijk in een vroegtijdig stadium van het ontwikkelings- en testproces potentieel kostbare fouten worden gevonden. De testbasis vormt de blauwdruk van het nieuw te bouwen systeem. Alles wat niet in de testbasis staat vermeld, wordt ter oplossing overgelaten aan het ontwikkelteam. Het ontwikkelteam gaat op basis van de systeemdocumentatie aan de slag om het nieuwe informatiesysteem te ontwikkelen. In deze documentatie kunnen fouten voorkomen die bij het niet tijdig ontdekken ervan veel en vaak kostbaar correctiewerk kunnen veroorzaken. Hoe eerder een fout in een ontwikkelingsproces wordt gevonden, hoe eenvoudiger (en goedkoper) een fout kan worden hersteld [Boehm, 1981]. Indien bijvoorbeeld een fout in een specificatie of requirement pas wordt ontdekt tijdens het uitvoeren van de acceptatietest zijn de herstelkosten groot. Niet alleen de software moet worden aangepast, maar bijvoorbeeld ook het technisch ontwerp en het functioneel ontwerp.

In het algemeen blijkt door vroege foutdetectie 50%-80% besparing van de testkosten mogelijk. Door het beoordelen van de testbasis zal, door een vroege foutdetectie, de kwaliteit van de testbasis toenemen.

Praktijkvoorbeeld

In onderstaande real-world voorbeelden werd de detailintake als onderdeel van het toetsen uitgevoerd:

- Een leverancier van pakketten heeft een ‘return-on-investment’ van 10:1 bereikt door het vroegtijdig toetsen van de ontwerpen. Hierdoor wordt €21,4 miljoen per jaar aan projectkosten bespaard en is de gemiddelde ‘time-to-market’ van de projecten met 1,8 maanden afgenomen.
- Een bedrijf in de telecomsector voorkomt door het toetsen van de code 33 uur aan herstelwerkzaamheden per ontdekte fout.
- Een grote computerfabrikant bespaart voor elk uur besteed aan inspecties 20 uur aan testinspanning en 82 uur aan herstelwerkzaamheden.
- Een multinational in de chemiesector geeft aan 400 geïnspecteerde softwareproducten 10x minder geld uit aan onderhoudskosten dan aan 400 niet-geïnspecteerde softwareproducten.

Context

Hoewel in het testplan zowel de (definitie van de) testbasis als de afgesproken teststrategie is vastgelegd, is op het moment van het opstellen van dit testplan de testbasis vaak nog niet beschikbaar. In de fase Voorbereiding moet worden onderzocht of de opgeleverde testbasis overeenstemt met en bruikbaar is voor de eerder vastgelegde afspraken in het plan. Als dit niet het geval blijkt te zijn, kan het nodig zijn het plan bij te sturen. Dit kan zowel een negatieve als een positieve invloed hebben op één of alle geld-, tijd- en kwaliteitsaspecten.

Negatieve invloed door bijvoorbeeld:

- het ontbreken van de definitieve testbasis een kwalitatief onvoldoende testbasis
- een testbasis met meer complexe algoritmes dan verwacht.

Positieve invloed door bijvoorbeeld:

- een testbasis met minder complexe algoritmes dan verwacht
- een testbasis die anticipeert op het maken van logische testgevallen (zie tip).

Het bijsturen van het plan is een activiteit uit de “Fase Beheer” (paragraaf 6.3) en wordt daar verder toegelicht.

Tip

In een overheidsinstelling werd besloten om de ontwerpers het functioneel ontwerp standaard aan te laten vullen met beslissingstabellen. Het idee hierachter was dat de ontwerpers zelf beter wisten wat zij met het ontwerp bedoelden dan de testers die op basis van het ontwerp de (logische) testgevallen moesten maken. Doordat de testers hierdoor een ‘vliegende start’ kregen en er minder hoefde te worden uitgezocht werd er 25% bespaard op de doorlooptijd van de fase Specificatie.

Randvoorwaarden

De fase Voorbereiding start zo vroeg mogelijk na fixatie van het testplan én na het beschikbaar zijn van de gefixeerde testbasis (zie uitgediept).

Uitgediept

De testbasis is gefixeerd als de opdrachtgever van de test aangeeft dat er voldoende activiteiten zijn uitgevoerd die de kwaliteit van de specificaties en overige informatie waarborgen. De fixatie van de specificaties is van groot belang. Ze vormen immers de basis voor zowel de testers als de ontwikkelaars en mogen alleen nog via formele wijzigingsprocedures veranderen.

Hoewel fixeren van de testbasis in principe alleen door de opdrachtgever kan worden gedaan, zijn er situaties denkbaar dat de testmanager overweegt zelf te doen alsof een testbasis is gefixeerd. Bijvoorbeeld omdat de testmanager de testvoortgang niet wil belemmeren, of testers ‘in de beschikbaarheid’ dreigt te krijgen. Bij het nemen van een dergelijk besluit is het van groot belang om hierover heldere afspraken te maken met de opdrachtgever. De kans is immers groot dat de testbasis nog gaat wijzigen met als mogelijke consequentie dat al gemaakte testontwerpen aangepast moeten worden. Dit kan extra kosten en verlenging van de testdoorlooptijd tot gevolg hebben. In de afspraken met de opdrachtgever moet worden vastgelegd hoe hiermee wordt omgegaan om geen discussie achteraf te krijgen.

Tip

Synergie tussen toetsen in ontwikkel- en testproces

In sommige organisaties worden ontwerpspecificaties structureel getoetst alvorens aan een volgende ontwikkelfase wordt begonnen. Door de diverse aandachtspunten uit de fase Voorbereiding onderdeel te laten zijn van een dergelijke toetsing ontstaat een goede mate van synergie tussen de structurele toetsing en de testactiviteiten uit de fase Voorbereiding. In deze situatie nemen één of meer leden van het testteam deel aan het toetsproces. Zij nemen het aspect testbaarheid met betrekking tot de ontwerpspecificaties voor hun rekening. Ook kunnen testers het

initiatief nemen tot de introductie van een structureel toetsproces (zijn requirements bijvoorbeeld SMART[1] opgesteld) met gebruikmaking van toetstechnieken zoals in H15 “Toetstechnieken” beschreven. Toetsen wordt dan een integraal onderdeel van de testaanpak. Bij het uitvoeren van de toetsactiviteiten kan uiteraard gebruik worden gemaakt van de diverse checklists zoals beschreven bij de activiteit “Opstellen checklists” (paragraaf 6.5.2).

Werkwijze

Nadat de testbasis aan het testteam beschikbaar is gesteld, wordt gestart met de intake hiervan. Hierbij wordt eerst onderzocht of de in het testplan vermelde opsomming van de informatie waaruit de testbasis bestaat nog steeds juist is. Indien nodig wordt deze, in overleg met de opdrachtgever, geactualiseerd. Tijdens dit onderzoek kan blijken dat (nog) niet alle informatie voor de tester beschikbaar is of misschien zelfs helemaal niet zal komen. In een dergelijke situatie moet een manier bedacht worden om aan de ontbrekende informatie te komen.

Als de testbasis duidelijk is wordt deze vanuit testperspectief beoordeeld op bijvoorbeeld consistentie, begrijpelijkheid en volledigheid. Vervolgens wordt aan de hand van checklists beoordeeld in hoeverre de vastgestelde teststrategie en de daaraan gerelateerde test(ontwerp)technieken hierop toepasbaar zijn. De conclusies worden vastgelegd in een rapport detailintake en met de opdrachtgever besproken. De resultaten van dit rapport kunnen aanleiding geven tot aanpassingen in de testbasis, de teststrategie en de te gebruiken testtechnieken.

Rollen / verantwoordelijkheden

Het opstellen van het rapport detailintake wordt door de testmanager of testcoördinator gedaan. Alle overige activiteiten kunnen door alle testteamleden uitgevoerd. Het rapport is bestemd voor de opdrachtgever van de test.

Activiteiten

De fase Voorbereiding bestaat uit de volgende activiteiten:

1. Verzamelen testbasis;
2. Opstellen checklists;
3. Beoordelen testbasis;
4. Opstellen rapport detailintake.

Onderstaand schema geeft de volgorde en afhankelijkheden aan tussen de verschillende activiteiten (figuur 6.16):

Figuur 6.16: Fase Voorbereiding.

6.5.1 Verzamelen testbasis

Doel

Het verzamelen van de definitieve, eventueel geactualiseerde, testbasis wordt in overleg met de opdrachtgever van de test uitgevoerd.

Werkwijze

De definitie van de relevante informatie voor het uitvoeren van de test is in principe al vastgelegd in het testplan (bijvoorbeeld functionele- en technisch ontwerpen, requirements, use cases, gebruikershandleidingen, interviewverslagen, prototype en referentiesysteem). Het is echter mogelijk dat ten aanzien van de uitgangsinformatie wijzigingen hebben plaatsgevonden. Het testplan dient dan te worden aangepast en de identificatie van de informatie moet worden herzien. Tenslotte worden de diverse onderdelen van de testbasis daadwerkelijk verzameld. Uiteindelijk dient het testteam uiteraard te beschikken over de juiste (versie van de) testbasis.

Aandachtspunt hierbij is dat de testbasis niet altijd aanwezig, compleet, actueel of in documenten vastgelegd hoeft te zijn. Een testbasis blijkt vaak incompleet te zijn doordat bijvoorbeeld niet-functionele requirements niet zijn gespecificeerd, terwijl deze wel als risicovol worden gezien. Door het project hierop te wijzen ontstaat er een (vroege) trigger om hier aandacht aan te schenken.

Alternatieve testbasis

Blijken er inderdaad testbasisproblemen te zijn dan volgen hier enkele aanpakken uit de praktijk om tot een alternatieve testbasis te kunnen komen:

- Huidige systeem in productie als referentiesysteem
  Stel dat de systeemdocumentatie ontbreekt, verouderd of niet compleet is. Bijvoorbeeld in een conversie- of migratieproject. Het maken, aanvullen of actualiseren van deze documentatie behoort meestal niet tot de ‘scope’ van het project. In een dergelijke situatie kan de huidige productieversie van het systeem als testbasis worden gebruikt. Dit is vooral een goed alternatief in situaties dat het project geen of nauwelijks wijzigingen op de functionele werking van het systeem met zich meebrengt, of als deze wijzigingen goed zijn gedocumenteerd.
- Prototype als testbasis
  In een situatie dat het maken van systeemdocumenten niet de hoogste prioriteit heeft en deze wellicht pas aan het einde van het project worden opgeleverd, wordt soms een prototype gemaakt. Dit komt bijvoorbeeld voor bij Rapid Application Development of varianten hierop (o.a. SP, DSDM, RUP). Aangezien het prototype vaak in samenwerking met de gebruiker wordt gemaakt, zou dit ook als testbasis gebruikt kunnen worden.
- Infosessie
  In bijvoorbeeld onderhoudstrajecten blijkt vaak dat zowel het systeem in productie als de wijzigingen hierop niet goed zijn gedocumenteerd. Het organiseren van een infosessie met alle betrokkenen (ontwikkelaars, ontwerpers, gebruikers, beheerders, enz.) blijkt in de praktijk een goed middel te zijn om duidelijkheid te krijgen over zowel de werking van bepaalde systeemdelen als over de door te voeren wijzigingen. De tijdens de infosessie verkregen informatie kan als testbasis worden gebruikt.
- Systeemdocumentatie uit voorlaatste iteratie als testbasis
  Bij iteratieve en incrementele systeemontwikkelaanpakken bestaat de mogelijkheid dat de systeemdocumentatie pas laat voor de tester beschikbaar komt. In de situatie dat de systeemdocumentatie tijdens de laatste iteratie niet meer mag worden gewijzigd, komt de testbasis aan het eind van de voorlaatste iteratie beschikbaar voor de tester. In de situatie dat de systeemdocumentatie tijdens de laatste iteratie wel mag worden gewijzigd, kan overwogen worden om de systeemdocumentatie (vaak voor meer dan 80% gereed) uit de voorlaatste iteratie als testbasis te gebruiken. Aan het eind van de laatste iteratie moeten dan nog de, vaak kleine, wijzigingen op de systeemdocumentatie door de tester in de testgevallen worden verwerkt.

Een belangrijk punt bij een op bovenstaande wijze verkregen alternatieve testbasis is dat deze door de opdrachtgever van de test (en eventuele andere ‘stakeholders’) wordt gezien als dé testbasis. Een op deze wijze verkregen testbasis zal echter zelden worden geaccordeerd of gefixeerd. Het is daarom van belang dat de opdrachtgever en de tester zich bewust zijn van de risico’s die dit met zich meebrengt. Het is verstandig om deze risico’s niet alleen te inventariseren maar ook om de bijbehorende maatregelen vast te leggen. Wie heeft er bijvoorbeeld een ‘beslissende stem’ als blijkt dat de gerealiseerde functionaliteit van een (deel)systeem anders is dan verwacht op basis van de alternatieve testbasis?

Soms is er zo weinig informatie aanwezig dat zelfs het vaststellen van een alternatieve testbasis niet lukt. In een dergelijke situatie kan worden gekozen voor andere informatiebronnen, die weliswaar niet als alternatieve testbasis te gebruiken zijn, maar wel goed bruikbaar zijn om bijvoorbeeld logische testgevallen van af te leiden (zie tips “Ontbreken van testbasis”).

Tips

Ontbreken van testbasis

Als er geen testbasis aanwezig is moet de tester op zoek gaan naar andere informatiebronnen die als basis voor het maken van testgevallen kunnen dienen. Bach, Whittaker en Kaner hebben hiervoor een eigen aanpak uitgewerkt:

- HICCUPP [Bach, 2003]
  Informatie voor het maken van testgevallen kan bijvoorbeeld worden gehaald uit normen en standaards, memo's, gebruikershandleidingen, interviews, advertenties of concurrerende producten. Bach heeft dit uitgewerkt in zijn HICCUPP aanpak:
  History. Is de huidige werking van de software consistent met de vorige?
  Image. Is de werking van de software consistent met de beeldvorming van de organisatie?
  Comparable. Is de werking van de software consistent met die van andere vergelijkbare producten?
  Claims. Is de werking van de software consistent met wat mensen zeggen zoals het zou moeten werken?
  User expectations. Is de werking van de software consistent met wat wij (testers) denken wat de gebruiker wil?
  Product. Is de werking van bepaalde softwarecomponenten consistent met vergelijkbare softwarecomponenten binnen het product?
  Purpose. Is de werking van de software consistent met het ogenschijnlijke doel van de software?

- 18 Attacks van Whittaker en Jorgenson [Whittaker, 2000]
  Sommige softwarefouten zijn zo triviaal dat daar goede standaard tests (attacks) voor te definiëren zijn. De hieronder genoemde 18 attacks van Whittaker en Jorgenson kunnen een prima basis vormen voor het maken van tests of als aanvulling worden gebruikt op al aanwezige tests:

  User interface (invoer)
  1. Genereer invoer waardoor alle foutboodschappen worden geforceerd.
  2. Genereer invoer waardoor alle default waarden moeten worden ingevuld.
  3. Probeer alle toegestane tekens en datatypen in te voeren.
  4. Voer te veel tekens in.
  5. Vind relaties tussen invoervelden en test combinaties van hun waarden.
  6. Voer herhaaldelijk dezelfde gegevens in.

  User interface (uitvoer)
  7. Probeer alle mogelijke uitvoer voor elke invoer uit.
  8. Probeer foute uitvoer te veroorzaken.
  9. Probeer eigenschappen/waarden van de uitvoer te wijzigen.
  10. 'Refresh' het scherm.

  Opgeslagen data
  11. Voer gegevens in vanuit alle mogelijke uitgangssituaties.
  12. Probeer in de database te veel of te weinig karakters op te laten slaan.
  13. Probeer alternatieve manieren te vinden om interne datarestricties te wijzigen.

  Berekeningen
  14. Probeer onjuiste operand- en operatorcombinaties uit.
  15. Probeer een rekenmodule zichzelf te laten aanroepen.
  16. Probeer de waarden van uitkomsten te hoog of te laag te laten zijn.
  17. Probeer functies te vinden die gebruik maken van dezelfde data.

  System interface (media)
  18a. Zorg dat er geen opslagruimte meer beschikbaar is.
  18b. Zorg dat het systeem bezig of niet beschikbaar is.
  18c. Beschadig het systeem.

  System interface (bestanden)
  18d. Ken een verkeerde bestandsnaam toe.
  18e. Wijzig rechten (o.a. lees- en schrijf rechten) van een bestand.
  18f. Wijzig de inhoud van een bestand of maak deze corrupt.

- 480 bugs van Kaner [Kaner, 1999]

Kaner heeft een lijst opgesteld met veel voorkomende softwarefouten. Deze lijst kan worden gebruikt om dezelfde of soortgelijke fouten in de te testen software te vinden. Of deze lijst kan in meer algemene zin worden gebruikt voor:

Opdoen van testideeën
Onderzoek of een fout uit de lijst ook in de te testen software kan optreden. Als dit theoretisch gezien mogelijk is, bedenk dan hoe je deze zou kunnen vinden. Maak vervolgens wel/geen testgevallen afhankelijk van de schade die de fout in productie zou kunnen veroorzaken.

Testontwerp review
Kies enkele testsituaties uit het testontwerp en zoek voor elke testsituatie naar een mogelijke fout in de lijst. Onderzoek vervolgens voor elke mogelijke fout of deze ook bij de te testen software kan optreden en of deze dan door de gemaakte testgevallen gevonden zou worden.

Blikverruiming
Zoek in de lijst naar type fouten waar vaak niet aan wordt gedacht (‘out of the box’ denken).

Opleiding
Laat nieuwe testers zien wat er fout kan gaan en laat ze testgevallen maken waarmee deze fouten gevonden kunnen worden.

Tips

Ontbreken van testbasis

Als er geen testbasis aanwezig is moet de tester op zoek gaan naar andere informatiebronnen die als basis voor het maken van testgevallen kunnen dienen. Bach, Whittaker en Kaner hebben hiervoor een eigen aanpak uitgewerkt:

- HICCUPP [Bach, 2003]
  Informatie voor het maken van testgevallen kan bijvoorbeeld worden gehaald uit normen en standaards, memo’s, gebruikershandleidingen, interviews, advertenties of concurrerende producten. Bach heeft dit uitgewerkt in zijn HICCUPP aanpak:
  History. Is de huidige werking van de software consistent met de vorige?
  Image. Is de werking van de software consistent met de beeldvorming van de organisatie?
  Comparable. Is de werking van de software consistent met die van andere vergelijkbare producten?
  Claims. Is de werking van de software consistent met wat mensen zeggen zoals het zou moeten werken?
  User expectations. Is de werking van de software consistent met wat wij (testers) denken wat de gebruiker wil?
  Product. Is de werking van bepaalde softwarecomponenten consistent met vergelijkbare softwarecomponenten binnen het product?
  Purpose. Is de werking van de software consistent met het ogenschijnlijke doel van de software?

- 18 Attacks van Whittaker en Jorgenson [Whittaker, 2000]
  Sommige softwarefouten zijn zo triviaal dat daar goede standaard tests (attacks) voor te definiëren zijn. De hieronder genoemde 18 attacks van Whittaker en Jorgenson kunnen een prima basis vormen voor het maken van tests of als aanvulling worden gebruikt op al aanwezige tests:

User interface (invoer)

1. Genereer invoer waardoor alle foutboodschappen worden geforceerd.
2. Genereer invoer waardoor alle default waarden moeten worden ingevuld.
3. Probeer alle toegestane tekens en datatypen in te voeren.
4. Voer te veel tekens in.
5. Vind relaties tussen invoervelden en test combinaties van hun waarden.
6. Voer herhaaldelijk dezelfde gegevens in.

User interface (uitvoer)

7. Probeer alle mogelijke uitvoer voor elke invoer uit.
8. Probeer foute uitvoer te veroorzaken.
9. Probeer eigenschappen/waarden van de uitvoer te wijzigen.
10. ‘Refresh’ het scherm.

Opgeslagen data

11. Voer gegevens in vanuit alle mogelijke uitgangssituaties.
12. Probeer in de database te veel of te weinig karakters op te laten slaan.
13. Probeer alternatieve manieren te vinden om interne datarestricties te wijzigen.

Berekeningen

14. Probeer onjuiste operand- en operatorcombinaties uit.
15. Probeer een rekenmodule zichzelf te laten aanroepen.
16. Probeer de waarden van uitkomsten te hoog of te laag te laten zijn.
17. Probeer functies te vinden die gebruik maken van dezelfde data.

System interface (media)

18a. Zorg dat er geen opslagruimte meer beschikbaar is.
18b. Zorg dat het systeem bezig of niet beschikbaar is.
18c. Beschadig het systeem.

System interface (bestanden)

18d. Ken een verkeerde bestandsnaam toe.
18e. Wijzig rechten (o.a. lees- en schrijf rechten) van een bestand.
18f. Wijzig de inhoud van een bestand of maak deze corrupt.

- 480 bugs van Kaner [Kaner, 1999]
  Kaner heeft een lijst opgesteld met veel voorkomende softwarefouten. Deze lijst kan worden gebruikt om dezelfde of soortgelijke fouten in de te testen software te vinden. Of deze lijst kan in meer algemene zin worden gebruikt voor:
  Opdoen van testideeën
  Onderzoek of een fout uit de lijst ook in de te testen software kan optreden. Als dit theoretisch gezien mogelijk is, bedenk dan hoe je deze zou kunnen vinden. Maak vervolgens wel/geen testgevallen afhankelijk van de schade die de fout in productie zou kunnen veroorzaken.
  Testontwerp review
  Kies enkele testsituaties uit het testontwerp en zoek voor elke testsituatie naar een mogelijke fout in de lijst. Onderzoek vervolgens voor elke mogelijke fout of deze ook bij de te testen software kan optreden en of deze dan door de gemaakte testgevallen gevonden zou worden.
  Blikverruiming
  Zoek in de lijst naar type fouten waar vaak niet aan wordt gedacht (‘out of the box’ denken).
  Opleiding
  Laat nieuwe testers zien wat er fout kan gaan en laat ze testgevallen maken waarmee deze fouten gevonden kunnen worden.

Bij gebruik van één of meer van genoemde aanpakken, om te komen tot een alternatieve testbasis of tot een basis om testgevallen van af te leiden, is het goed dat de tester zich realiseert dat het niet de taak van de tester is om de testbasis te maken. De tester beoordeelt en gebruikt de testbasis uitsluitend voor testdoeleinden. Het maken van systeemdocumenten was, is en blijft de verantwoordelijkheid van bijvoorbeeld het project of de ontwikkelafdeling. De tester moet niet op de stoel van de ontwerper gaan zitten. Dit betekent dat de testbasis die uit één van de genoemde aanpakken wordt verkregen te allen tijde met alle betrokkenen moet worden afgestemd. Enerzijds om zeker te stellen dat het systeem daadwerkelijk zo moet functioneren en/of moet worden gebouwd, anderzijds om zeker te stellen dat men ermee instemt dat dit inderdaad de alternatieve testbasis is waartegen moet worden getest, of de basis is waar testgevallen van moeten worden afgeleid.

Producten

Gefixeerde testbasis.

Technieken

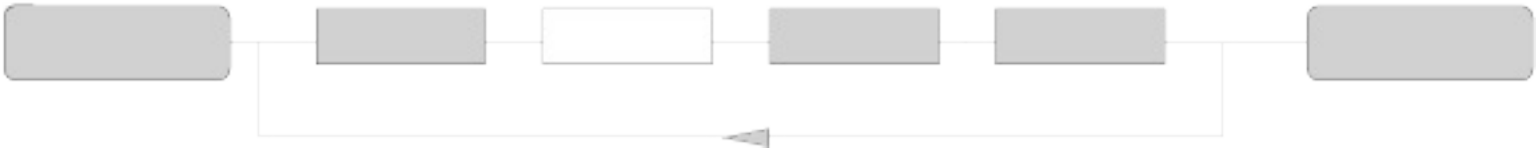
HICCUPP [Bach, 2003].
18 Attacks van Whittaker en Jorgenson [Whittaker, 2000].

480 Bugs van Kaner [Kaner, 1999].

Tools

Niet van toepassing.

6.5.2 Opstellen checklists



Doel

Het opstellen van checklists, aan de hand van de in het testplan vastgestelde teststrategie, voor de verschillende te testen deelobjecten/kenmerken. Deze checklists vormen de leidraad voor het beoordelen van de testbasis.

Werkwijze

Met behulp van checklists wordt de testbasis gecontroleerd op testbaarheid.

In deze activiteit worden de daarvoor benodigde checklists opgesteld. Afhankelijk van de geselecteerde testontwerptechnieken, testvormen, de informatiebronnen die de testbasis bepalen en de te testen deelobjecten/kenmerken dienen één of meer checklists te worden opgesteld (zie ook tip “Testontwerptechnieken bij ontbreken van testbasis”). In elke checklist moet worden aangegeven welke specifieke controleaspecten bij de intake een rol spelen. Als men wil voorkomen dat voor de beoordeling van identieke delen van de testbasis deze meerdere malen moeten worden doorlopen, kunnen de afzonderlijke checklists worden samengevoegd tot één checklist. Bij het opstellen van de checklist kan gebruik worden gemaakt van de checklists “testontwerptechnieken”, zoals deze op www.tmap.net is te vinden.

Mede gezien de diversiteit aan testontwerptechnieken en informatiebronnen die de testbasis bepalen, is het niet mogelijk één algemeen geldende checklist per deelobject/kenmerk op te stellen. Per organisatie en per project dienen daarom specifiek op de situatie toegesneden checklists te worden opgesteld. Het verdient aanbeveling om altijd een checklist op te stellen, omdat in de praktijk blijkt dat zonder een checklist vaak ‘te veel’ of juist uitsluitend op het gebruik van standaards en correcte spelling wordt gelet. Dit kan bij de diverse betrokkenen onnodig wrijving veroorzaken.

Tip

Testontwerptechnieken bij ontbreken van testbasis

In het testplan zijn onder andere de opsomming van de informatie waaruit de testbasis bestaat en de teststrategie vastgelegd. Als echter blijkt dat de afgesproken (gedocumenteerde) testbasis gedeeltelijk of in het geheel ontbreekt, kan het zijn dat er op basis van andere (niet gedocumenteerde) informatie moet worden getest. In dat geval zijn niet alle testontwerptechnieken geschikt.

Enkele testontwerptechnieken die in een dergelijke situatie vaak wel toepasbaar zijn:

- Datacombinatietest (DCT), ook bekend onder de naam classification tree method (CTM)
- Error guessing
- Exploratory testing
- Grenswaardenanalyse
- Afvinklijst.

Voor toelichting op en gebruik van deze testontwerptechnieken wordt verwezen naar hoofdstuk 14 “Testontwerptechnieken”.

Producten

Diverse checklists of één geaggregeerde checklist voor het beoordelen van de testbasis.

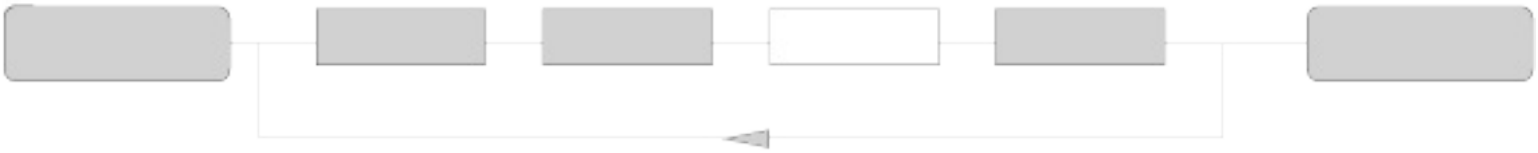
Technieken

Checklist “testontwerptechnieken” (www.tmap.net).

Tools

Niet van toepassing.

6.5.3 Beoordelen testbasis



Doel

Het vaststellen van de testbaarheid van de testbasis. Met testbaarheid wordt hier bedoeld de volledigheid, consistentie, toegankelijkheid en vertaalbaarheid naar testgevallen.

Werkwijze

De testbasis wordt beoordeeld met behulp van toetstechnieken en aan de hand van de opgestelde checklist(s) om inzicht te krijgen in de toepasbaarheid van de vastgestelde teststrategie en de daaraan gerelateerde testontwerptechnieken. Als blijkt dat de testbasis niet voldoet, is het uiteraard van belang hierover zo snel mogelijk via de opdrachtgever te rapporteren aan de leverancier van de testbasis. Deze kan dan zorg dragen voor opheldering van de onduidelijkheden en/of invulling van de hiaten. De registratie en aanmelding van deze bevindingen op de testbasis vinden plaats door middel van de in paragraaf 6.2.12 “Inrichten beheer” vastgestelde procedures.

Producten

Testbasisbevindingen.

Technieken

Checklist voor beoordelen testbasis (product uit 6.5.2).

Toetstechnieken (hoofdstuk 15).

Tools

Bevindingenbeheertool.

6.5.4 Opstellen rapport detailintake

Doel

Het rapport detailintake:

- geeft een terugkoppeling over de kwaliteit van de testbasis en de impact hiervan op het geplande testtraject
- stelt zwakke plekken in het systeemontwerp vroegtijdig ter discussie
- informeert over projectrisico's.

Werkwijze

Op basis van de individuele testbasisbevindingen wordt een rapport detailintake opgesteld. Dit rapport geeft een algemene samenvatting ten aanzien van de kwaliteit c.q. de testbaarheid van de testbasis. Eventuele consequenties van onvoldoende kwaliteit dienen eveneens te worden beschreven. Tevens worden verschillen met de in het testplan vastgelegde opsomming van de informatie waaruit de testbasis bestaat en de afgesproken teststrategie beschreven. Dit kan aanleiding geven tot het 'bijsturen' van het plan met betrekking tot bijvoorbeeld de te volgen strategie en de te gebruiken testtechnieken. Voor verdere toelichting hierop wordt verwezen naar de "Fase Beheer" (paragraaf 6.3).

Het rapport detailintake kan bijvoorbeeld uit de volgende paragrafen bestaan:

- Opdrachtformulering
  Een identificatie van de oorspronkelijke (of eventueel aangepaste) testbasis evenals een beschrijving van de opdrachtgever en de opdrachtnemer.
- Conclusie
  De eindconclusie met betrekking tot de testbaarheid van de onderzochte testbasis en eventueel hiermee samenhangende consequenties c.q. risico's: is de testbasis van voldoende kwaliteit om zinvol te kunnen starten met het specificeren van tests zoals vastgelegd in de (eventueel aangepaste) teststrategie?
- Aanbevelingen
  Aanbevelingen met betrekking tot de beoordeelde testbasis en eventuele structurele aanbevelingen om in de toekomst een betere testbasis te kunnen opleveren.
- Bevindingen
  De gedane bevindingen worden in detail beschreven of er wordt verwezen naar de bijbehorende bevindingenformulieren.
- Bijlagen
  De gebruikte checklists.

Producten

Rapport detailintake.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.6 Fase Specificatie

Doel

Het specificeren van de benodigde tests en uitgangssituatie(s). Het doel is zoveel mogelijk voorbereid te hebben om de testuitvoering zo snel mogelijk te laten verlopen wanneer de ontwikkelaars het testobject opleveren.

Context

Deze fase start wanneer de detailintake op de testbasis is uitgevoerd en de bevindingen hierop zoveel mogelijk zijn verwerkt. De testspecificatie loopt parallel aan en ook in de luwte van de realisatie van de software (of parametrisering in geval van pakketten). De software is het primaire product van het ontwikkelproces én ligt normaal gesproken op het kritieke pad van het proces. De focus van het (project)management is daarom hierop gericht. De testspecificatie heeft slechts zijdelingse belangstelling. Dit verandert op het moment dat de software overgedragen wordt voor testuitvoering. Op dat moment komt de aandacht van (project)management daarop te liggen. Het testteam moet dan klaar staan om met testuitvoering te starten. De testspecificatie is er dan ook op gericht om zoveel mogelijk voorbereid te hebben zodat de testuitvoering zo snel mogelijk kan verlopen en zo kort mogelijk op het kritieke pad verblijft.

De testmanager moet zich hiervan bewust zijn. Hij moet de signalen dat de testspecificatie problemen ondervindt, zoveel mogelijk vertalen naar gevolgen (in tijd, geld en kwaliteit) voor latere testuitvoering en het totale voortbrengingsproces.

Randvoorwaarden

Aan de volgende voorwaarden dient te zijn voldaan alvorens kan worden gestart met de fase Specificatie:

- de testbasis is beschikbaar en staat onder configuratiebeheer;
- de bevindingen uit de detailintake testbasis zijn verwerkt.

Werkwijze

Tijdens de specificatiefase specificeren de testers per testeenheid de benodigde tests. Dit gebeurt door het opstellen van checklists of het specificeren van testge vallen aan de hand van de toegewezen testontwerptechnieken. In dit laatste geval stellen de testers ook testscripts op waarin de testgevallen in een efficiënt uitvoerbare volgorde worden gezet. Op basis hiervan en deels parallel hieraan definiëren de testers één of meer centrale uitgangssituaties voor het testen, waar de testgevallen gebruik van kunnen maken. Dit kan een kopie van productie of een centrale tabelvulling zijn. Een speciale vorm van een te specificeren test is de intake op het testobject. Deze test moet in de fase Uitvoering controleren of het testobject voldoende testbaar is voor een zinvolle en efficiënte testuitvoering.

Rollen/verantwoordelijkheden

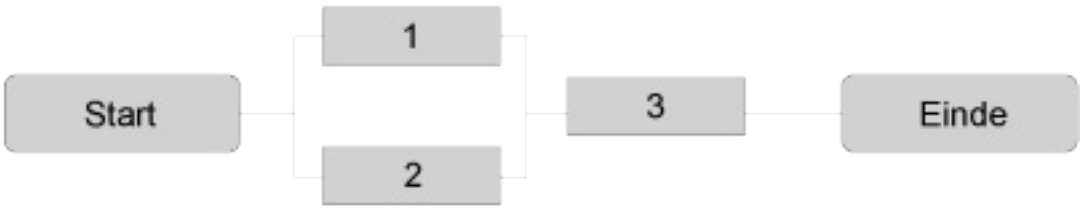
De activiteiten in de fase Specificatie worden uitgevoerd door de testers.

Activiteiten

Binnen de fase Specificatie worden de volgende activiteiten onderkend:

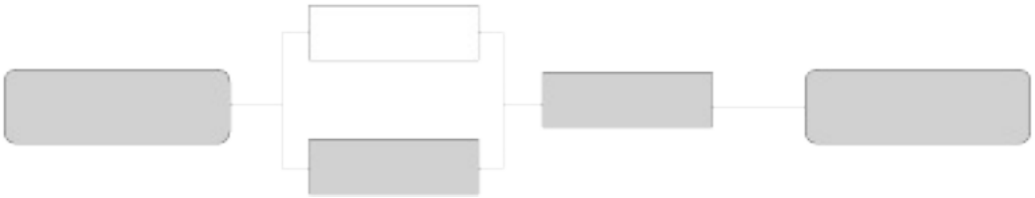
1. Opstellen testspecificaties;
2. Definiëren centrale uitgangssituatie(s);
3. Specificeren intake testobject;

Onderstaand schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten. Activiteiten 1 en 2 beïnvloeden elkaar over en weer.



Figuur 6.17: Fase Specificatie.

6.6.1 Opstellen testspecificaties



Doel

Het per testeenheid opstellen van de testspecificaties.

Werkwijze

Voor de testeenheden uit het testplan specificeren de testers de benodigde tests. Na afronding worden de testspecificaties in beheer genomen.

Definitie

Een testeenheid is een verzameling processen, transacties en/of functies die gezamenlijk worden getest.

Afhankelijk van de voor de testeenheid gekozen testvorm en -techniek kan deze activiteit bestaan uit het opstellen van een checklist tot het ontwerpen en specificeren van testgevallen volgens een testontwerptechniek of tot het ontwerpen van een test met andere technieken. De mogelijkheden worden hieronder verder toegelicht. Ook worden toelichtingen gegeven op een schaalbare regressietest en op de relatie tussen deze fase en exploratory testing.

Gedurende deze activiteit kunnen er problemen ontstaan met de gehanteerde testbasis. Ruwweg zijn deze in te delen in de volgende categorieën:

- Bevindingen
  Net als bij de detailintake kunnen de testers tekortkomingen en/of onduidelijkheden constateren in de testbasis. De testers stellen hiervoor een bevinding op. Via de bevindingenprocedure komt deze bij de leverancier van de testbasis, waarna deze de bevinding kan oplossen.
- Ontbreken van testbasis
  Wanneer de detailintake onvoldoende is uitgevoerd, kan nu pas blijken dat bepaalde onderdelen van de testbasis ontbreken of niet gedetailleerd genoeg zijn, waardoor ze niet of onvoldoende testbaar zijn. Dezelfde soorten maatregelen als bij de detailintake kunnen overwogen worden, zie paragraaf 6.5 “Fase Voorbereiding”.

Tip

Bij iteratieve of agile systeemontwikkeling is de testbasis aan het begin van de iteratie vaak niet 100% compleet, maar wordt deze gedurende de iteratie gecompleteerd. Naast bovengenoemde maatregelen is de aanbeveling om bij elke aanvulling van de testbasis een minimale detailintake uit te voeren alvorens tests te gaan specificeren op basis van de aanvulling.

- Instabiele testbasis
  Wanneer de leverancier van de testbasis deze tijdens het specificeren van de tests nog regelmatig wijzigt, bijvoorbeeld als gevolg van bevindingen of wijzigingsvoorstellen, is sprake van een instabiele testbasis. Bij elke wijziging moeten de testers de relevante testspecificaties bekijken of er aanpassingen nodig zijn. Deze herstelwerkzaamheden zijn vooraf altijd erg moeilijk in te schatten. De testmanager doet er goed aan hiervoor bij het opstellen van het testplan een bepaalde hoeveelheid reservebudget en -tijd te reserveren. Bij overschrijding hiervan dient geëscaleerd te worden naar projectmanagement dat meer tijd en geld nodig is (zie ook paragraaf 6.3 “Fase Beheer”). Andere mogelijke maatregelen zijn om het specificeren van de tests voor de instabiele delen in de planning naar achteren te schuiven of om de logische testgevallen al wel op te stellen, maar het fysiek maken ervan uit te stellen tot een later moment, wanneer de testbasis stabiel(er) is geworden.

Testontwerptechnieken

Voor het opstellen van testgevallen wordt gebruik gemaakt van testontwerptechnieken (zie hoofdstuk 14). Hiermee kunnen de testers op eenduidige, overdraagbare en reproduceerbare wijze uit een bepaalde testbasis testgevallen afleiden die een bepaalde dekking bereiken. Een testgeval bestaat uit een beschrijving van de initiële situatie, de testactie en de resultaatvoorspelling. Het specificeren van testgevallen volgens een testontwerptechniek doorloopt de volgende generieke stappen.

1. Identificeren testsituaties
2. Opstellen logische testgevallen
3. Opstellen fysieke testgevallen
4. Vaststellen uitgangssituatie
5. Opstellen testscript

De begrippen in deze stappen worden in meer detail toegelicht in paragraaf 14.2.1 “Testsituatie, testgeval en testscript”, inclusief voorbeelden. Hieronder volgt meer uitleg over de stappen zelf.

Voor het vastleggen van het resultaat van deze stappen, die gezamenlijk de testspecificatie vormen, moet een keuze gemaakt worden. Op hoofdlijnen zijn twee varianten mogelijk. De eerste is om alles in één bestand bij elkaar te voegen, vaak een tekstdocument of spreadsheet. De tweede is om een splitsing aan te brengen in de pure ontwerpstappen (identificeren testsituaties en opstellen logische testgevallen) en de stappen die nodig zijn voor testuitvoering (de overige, resulterend in het script). Het voordeel van het eerste is dat onderhoud op de testgevallen vergemakkelijkt wordt omdat alles bij elkaar staat. De tweede variant heeft dit voordeel niet, maar heeft twee andere voordelen. De uitvoerende tester (bijvoorbeeld een gebruiker die de test moet uitvoeren maar niet ontworpen heeft) wordt niet afgeleid door de ontwerpstappen maar ziet enkel de informatie die nodig is om de test uit te voeren. Het tweede voordeel is dat de voor uitvoering benodigde informatie in een andere vorm dan tekstverwerker of spreadsheet kan worden vastgelegd, zoals een testwarebeheertool of een database of rechtstreeks in een tool voor geautomatiseerde testuitvoering.

1. Identificeren testsituaties

Elke testontwerptechniek is gericht op het bereiken van een bepaalde dekking om bepaalde soorten fouten te vinden. Om deze te vinden is het onderkennen van de te testen situaties in de testbasis de eerste stap. De testbasis bestaat bijvoorbeeld uit de system requirements, het functionele ontwerp, de gebruikershandleiding en/of de administratieve procedures. Deze testbasis wordt doorlopen en elke te testen situatie wordt hierin geïdentificeerd. De testontwerptechniek schrijft bijvoorbeeld als te testen situaties voor dat van elke conditie de waar- en onwaar-situatie wordt getest, of dat elk invoerveld met een valide en invalide invoerwaarde wordt getest. Omdat het onderkennen van testsituaties voor iedere testontwerptechniek anders is, zitten in deze stap de grootste verschillen tussen de diverse testontwerptechnieken.

Uitgediept

Omvormen testbasis

In sommige gevallen is de testbasis niet rechtstreeks geschikt voor de gekozen testontwerptechniek, maar moet een extra omvormingsactie uitgevoerd worden. Zo heeft de techniek gegevenscyclustest een CRUD-matrix nodig en de beslistabellentechniek een beslistabel. Wanneer deze niet aanwezig zijn, kunnen ze vaak wel afgeleid worden uit de overige vormen van testbasis. Los van de vraag of de gekozen testontwerptechniek de juiste is (want soms is er weinig keus), is ook een vraag wie dit omvormen moet gaan doen. Bij voorkeur zijn dit de ontwerpers. Als de testers echter de enigen zijn die om deze aanvulling vragen, is de kans niet heel groot dat de ontwerpers dit op zich nemen. De andere mogelijkheid is dat de testers het zelf doen. Hierbij moeten zij opletten dat ze alleen omvormen en geen informatie toevoegen. In dat laatste geval zijn ze zelf het ontwerp aan het aanvullen en dat is niet de taak van de testers.

2. Opstellen logische testgevallen

In principe doorloopt een testgeval het betreffende deel van het testobject (bijvoorbeeld een functie) van begin tot eind. Daarbij dekt dat testgeval één of meerdere testsituaties af.

De logische testgevallen laten op een overzichtelijke manier zien:

- wat de testsituaties zijn;
- dat alle gedefinieerde testsituaties afgedekt zijn met testgevallen.

Het opstellen van logische testgevallen kan overgeslagen worden als de testsituaties rechtstreeks stuk voor stuk uitgevoerd en beoordeeld kunnen worden.

3. Opstellen fysieke testgevallen

Het fysieke testgeval beschrijft in praktische termen wat er gedaan moet worden, waarbij de drie basiselementen van een testgeval herkenbaar zijn: initiele situatie, actie en resultaatvoorspelling.

Bij het fysiek maken wordt elke te testen situatie of logisch testgeval zo ver gedetailleerd en concreet ingevuld dat later tijdens de testuitvoering zo efficiënt mogelijk gewerkt kan worden.

De keuze voor de fysieke invulling is vanzelfsprekend gebaseerd op de voorgaande stappen, maar wordt daarnaast beïnvloed door afspraken over bijvoorbeeld het gebruik van een bepaalde (centrale) uitgangssituatie, bijvoorbeeld een (geanonimiseerde) kopie van de productiedatabase.

Ook moeten afspraken gemaakt worden dat de verschillende tests elkaar niet kunnen verstoren tijdens de uitvoering, bijvoorbeeld omdat verschillende testgevallen dezelfde gegevens gebruiken (concurrentie). Als het ene testgeval “verwijderen orderregels” test en het andere juist “wijzigen orderregels”, dan is er grote kans dat de voorspelde resultaten ten onrechte niet gerealiseerd zullen worden. Een oplossing is bijvoorbeeld om voor de orderregels bepaalde series van nummers af te spreken ten behoeve van het testen van verschillende functies, zie ook paragraaf 6.6.2.

In dit verband is het ook aan te bevelen om geen afhankelijkheden tussen testgevallen te creëren. Wanneer de uitvoering van het ene testgeval voorwaardelijk is voor de uitvoering van het andere, betekent het foutlopen van het eerste testgeval ook dat het tweede testgeval niet uitgevoerd kan worden. Pas wanneer de bevinding die het eerste testgeval blokkeerde, is opgelost, kan het tweede testgeval doorlopen worden. Wanneer dit ook weer een bevinding oplevert, kan dit leiden tot een extra en onnodige hertestronde.

In paragraaf 14.2.1 “Testsituatie, testgeval en testscript” staat een aantal overwegingen hoe ver de tester moet gaan met het fysiek maken van testgevallen.

4. Vaststellen uitgangssituatie

Om een fysiek testgeval te kunnen uitvoeren, is een initiële situatie nodig. Vaak bevatten de initiële situaties voor meerdere testgevallen dezelfde gegevens. Dergelijke gegevens worden dan in een zogenaamde uitgangssituatie voor de gehele test opgenomen en niet meer afzonderlijk bij elk testgeval. Deze uitgangssituatie wordt voorafgaand aan de testuitvoering klaargezet.

Een stap verder is dat de uitgangssituaties voor verschillende tests ook (grote) overlap kunnen vertonen. Om die reden is ook vaak sprake van één of meer centrale uitgangssituaties die voor meerdere tests van toepassing zijn. Dit is uitgebreid beschreven in paragraaf 6.6.2.

5. Opstellen testscript

Als laatste stap worden één of meer testscripts opgeleverd. Hierbij zijn de testacties en controles van de fysieke testgevallen beschreven in een voor testuitvoering optimale volgorde. Hierbij moeten de testgevallen elkaar niet kunnen verstoren. Het testscript is als zodanig het stappenplan voor de testuitvoering en biedt tevens de mogelijkheid van voortgangsbewaking. De fysieke testgevallen en de uitgangssituatie vormen uiteraard de basis voor het te vervaardigen testscript. Overwegingen waarom testgevallen in één script worden samengenomen, zijn beschreven in paragraaf 14.2.1.

De generieke inhoudsopgave van een script is als volgt:

- Unieke identificatie, bestaande uit:
  - versie;
  - opsteller;
  - testbasis inclusief versie.

- Klaarzetten van de uitgangssituatie
  bijvoorbeeld door het zetten van de systeemdatum, het restoren van een bepaalde back-up en het toevoegen van bepaalde testgegevens
- Testacties en -controles
  De fysieke testgevallen in een voor uitvoering geschikte volgorde, met voor elk testgeval de benodigde initiële situatie, actie en resultaatcontrole. Wanneer een goede uitgangssituatie is neergezet, hoeft voor de initiële situatie meestal niets meer gedaan te worden.
- Opschonen omgeving
  Zorgen dat de resultaten van de uitgevoerde test zo nodig weer geschoond worden zodat andere testers hier geen verstoring van kunnen ondervinden (denk bijvoorbeeld ook aan het terugzetten van de systeemdatum).

Bovenstaande zegt niets over hoe het script wordt opgeslagen, bijvoorbeeld in een tekstdocument, spreadsheet, geautomatiseerde test of een database.

Paragraaf 14.2.1 geeft een aantal mogelijkheden en overwegingen.

Bij deze stappen is in sommige gevallen ondersteuning door tools mogelijk, bijvoorbeeld om testgegevens te genereren, om testgevallen af te leiden of zelfs model based testing-tools (zie paragraaf 8.5.3 “Soorten testtools”).

Checklists

Naast het specificeren van testgevallen vinden ook veel tests plaats met behulp van checklists. Deze worden gebruikt bij eenvoudige functionele tests, maar ook voor het statisch testen van bijvoorbeeld onderhoudbaarheid, beheerbaarheid, gebruikersvriendelijkheid of beveiliging. Hoewel een checklist meestal specifiek is voor de situatie, hanteren de testers vaak een algemene checklist als basis en maken hierop de specifieke aanpassingen. De algemene checklists kunnen vanuit de eigen organisatie beschikbaar zijn gesteld (door de testafdeling!) of uit de literatuur of via internet. Op www.tmap.net zijn diverse voorbeelden van checklists voor het testen van bepaalde kwaliteitsattributen te vinden.

Het opstellen (en uitvoeren) van een checklist vereist een bekwame tester die de nodige kennis heeft van het te testen deelobject of kenmerk. Het laten reviewen van de checklist is daarom aan te bevelen.

Overige technieken

Behalve het specificeren van testgevallen volgens testontwerptechnieken en het maken van checklists, zijn nog andere technieken mogelijk die niet in één van beide bovengenoemde categorieën vallen. Deze technieken zijn met name van toepassing op het testen van kwaliteitsattributen als portabiliteit, usability, performance en beveiliging. Voorbeelden hiervan worden gegeven in paragraaf 10.3 “Testvormen”.

Opbouwen en beheren schaalbare regressietests

Uitgediept

In de praktijk zijn regressietests vaak gebrekkig opgezet. In deze paragraaf wordt een aanpak beschreven voor het opbouwen, gebruiken en beheren van regressietests gebaseerd op het TestKubus principe [TestKubus, 2005]. Daarin worden samenhangende principes beschreven waarmee het mogelijk wordt om:

- testgevallen te specificeren en uit te voeren op basis van prioritering;
- snel en adequaat te rapporteren over de voortgang van testspecificatie en/of testuitvoering;
- testtrajecten nauwkeurig te plannen en te begroten;
- snel en variabel regressietests samen te stellen;
- wijzigingen in het testobject eenvoudig te verwerken in de test.

Het principe achter de TestKubus is dat per testgeval een verzameling aanvullende gegevens wordt vastgelegd: de testgevallen in de test worden ‘geclassificeerd’. Met behulp van deze classificaties kunnen langs allerlei dwarsdoorsneden subsets van testgevallen uit de totale test worden geselecteerd.
Voorbeelden van classificaties zijn:

- applicatie

- deelobject
- functie
- risicoklasse
- proces(onderdeel)
- release
- requirement
- transactie
- zwaartecategorie

Een goede selectie van classificaties en het correct classificeren van de testgevallen is bepalend voor de bruikbaarheid van dit concept. Essentieel hierbij is de classificatie naar zwaartecategorie. Deze classificatie geeft het 'gewicht' van het testgeval in de test aan en maakt het mogelijk om met een variabele diepgang een op risico's gebaseerde regressietest samen te stellen.

De toepassing van deze zwaartecategorieën bij het samenstellen van regressietests is als volgt (bij indeling in drie categorieën):

- Door van een deelobject alleen de testgevallen van categorie 1 te selecteren ontstaat een kleine regressietest. Deze subset wordt gebruikt voor een deelobject waarop geen aanpassingen zijn gedaan (of voor een intaketest op een nieuw of ingrijpend gewijzigd deelobject).
- De testgevallen van categorie 2 (= inclusief categorie 1) leveren een normale regressietest op, bijvoorbeeld voor een deelobject waarin aanpassingen zijn gedaan.
- De testgevallen van categorie 3 (= inclusief categorie 1 en 2) dekken het totale deelobject af en worden toegepast bij nieuwe of ingrijpende gewijzigde deelobjecten.

Aan de mate van detail waarin de testgevallen gespecificeerd worden, worden geen eisen gesteld. Wanneer wordt voorzien dat testgevallen uitgevoerd gaan worden door testers zonder materiekennis, dienen de testgevallen meer in detail te worden uitgeschreven.

Slechts op één punt stelt het concept een eigen specifieke eis aan de testgevallen. Ze moeten namelijk onafhankelijk van elkaar zijn, zoals beschreven bij het opstellen van de fysieke testgevallen. Dit heet het zogenaamde onafhankelijkheidsprincipe van het concept. Ook moeten de testgevallen parallel aan elkaar uitgevoerd kunnen worden. Testgevallen die voor een bepaalde periode exclusief gebruik van de testomgeving vereisen, belemmeren de uitvoering van andere testgevallen. Dit bemoeilijkt het plannen van de doorlooptijd van het testtraject.

Toepassing van dit concept maakt de omvang van de (regressie)test en de daarmee samenhangende activiteiten in het testtraject goed meetbaar.

Net als bij de testware in het algemeen moet ook hier goed worden nagedacht over wanneer, hoe en door wie deze test actueel gehouden kan worden.

Voor een nadere toelichting wordt naar de desbetreffende white-paper verwezen [TestKubus, 2005].

Session based exploratory testing

Uitgediept

Hoewel exploratory testing (ET) in hoofdstuk 14 meer in detail wordt besproken, is dit feitelijk geen pure testontwerptechniek. Bij ET maakt de tester tijdens de testuitvoering telkens een keuze welke test hij wil uitvoeren. Hij ontwerpt ter plekke een test, gebruikmakend van zijn kennis van testontwerptechnieken, zonder deze te documenteren. Als zodanig heeft ET in de fase Specificatie geen plaats, alles gebeurt immers tijdens testuitvoering. De reden om hier al wel aandacht aan te geven is dat om ET beter beheersbaar te maken, dit vaak in de vorm van sessies wordt georganiseerd met duidelijke testopdrachten die in enkele uren volbracht kunnen worden. Deze testopdrachten worden testcharters genoemd. Hoewel de lijst testcharters dynamisch is, doen de testers er goed aan om een eerste lijst testcharters op te stellen vóór de fase Uitvoering.

Voor meer informatie over ET en testcharters wordt verwezen naar paragraaf 14.4.6.

Producten

Bevindingen op de testbasis.
Testspecificaties (checklists, testgevallen, testscripts).

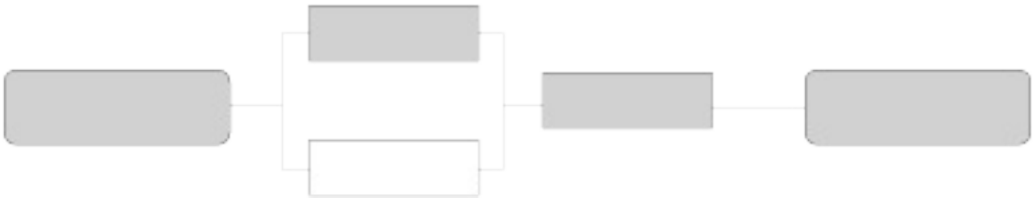
Technieken

Testontwerptechnieken (hoofdstuk 14).
Checklists voor diverse kwaliteitsattributen, www.tmap.net.

Tools

Testdatatool.
Testontwerptool.
'Model based testing'-tool.
Testwarebeheertool.
Geautomatiseerde testuitvoeringstool.

6.6.2 Definiëren centrale uitgangssituatie(s)



Doel

Het definiëren van één of meer centrale uitgangssituaties waar de testers gegevens voor hun testspecificaties uit kunnen halen.

Werkwijze

Een goede uitgangssituatie is van essentieel belang voor het kunnen (her)testen. Dit omvat alles wat nodig is om het testobject en de testomgeving in de toestand te krijgen dat gestart kan worden met de testgevallen in het testscript. Hieronder vallen niet alleen de testgegevens die voor de 'processing' nodig zijn, maar ook de toestand waarin het systeem en zijn omgeving zich in moeten bevinden. Denk bijvoorbeeld aan het instellen van een bepaalde systeemdatum, of het draaien van bepaalde week- en maandbatches die het systeem in een bepaalde toestand brengen.

In de praktijk blijken verkeerde uitgangssituaties een belangrijke bron van problemen voor het testen. Om te voorkomen dat tijdens de testuitvoering met de verkeerde uitgangssituatie wordt getest, moet daarom in een vroegtijdig stadium worden nagedacht over de manier waarop deze wordt opgebouwd en welk proces wordt gehanteerd bij het gebruik ervan. Wanneer dit niet gebeurt, kunnen de volgende problemen ontstaan:

- Niet reproduceerbare testresultaten
  Wanneer een testscript twee keer wordt uitgevoerd op dezelfde versie van het testobject en de resultaten verschillen, dan kan dit het gevolg zijn van afwijkende testgegevens in de uitgangssituatie. Mogelijk zijn voor andere tests extra gegevens toegevoegd of juist verwijderd in de uitgangssituatie.
- Uitgangssituatie wordt steeds slechter
  Tijdens de testuitvoering worden testgegevens gebruikt en aangepast. Nieuwe gegevens komen in het systeem, bestaande gegevens worden aangepast of misschien wel verwijderd. Wanneer er geen proces bestaat om de uitgangssituatie te beheren kan er niets meer gezegd worden over de kwaliteit ervan.
- Testen wordt steeds duurder
  Wanneer de uitgangssituatie van slechte kwaliteit is en nergens staat beschreven, dan moeten testers steeds meer moeite (door het zoeken of bedenken van testgegevens) doen om de testgevallen uit te voeren. Bovendien wordt de

kans steeds groter dat de tester een fout maakt. Dit zal zelfs groeien in de tijd omdat de uitgangssituatie steeds onbekender en daardoor slechter wordt.
- Onvoldoende informatie bij bevindingen zorgt voor vertraging
  Bij het rapporteren van een bevinding neemt de uitgangssituatie een belangrijke plaats in. Juist daarmee wordt een bevinding nog duidelijker. Wanneer deze uitgangssituatie tijdens het onderzoeken van de bevinding niet bekend is, zorgt dit voor vertraging. Ontwikkelaars moeten zelf op zoek gaan naar de oorspronkelijke uitgangssituatie of moeten om meer duidelijkheid vragen bij de tester.

In de testspecificaties worden per testscript de benodigde uitgangssituaties gespecificeerd. Om redundantie te voorkomen en het aantal benodigde fysieke bestanden te beperken, worden één en eventueel meerdere centrale uitgangssituaties gedefinieerd waar de testers bij het opstellen van hun testgevallen gebruik van kunnen maken.

Het samenstellen van centrale uitgangssituaties kan parallel aan het opstellen van de testspecificaties gebeuren en is vaak een iteratief proces. Vaak maakt een tester een begin met een centrale uitgangssituatie door bijvoorbeeld een vulling van stambestanden voor te stellen. Stambestanden zijn gegevens die het systeem sturen, maar geen onderdeel zijn van de primaire gegevensverwerking. Voorbeelden zijn kortingstabellen, belastingpercentages, postcodetabel, productsoorten en klantsoorten. Een volgende stap kan zijn om alvast een eerste vulling van primaire gegevens voor te stellen, bijvoorbeeld een aantal klanten, producten, orders en facturen. Er kan besloten worden om meerdere centrale uitgangssituaties te definiëren, wanneer dit handig lijkt voor het specificeren van de tests. Het verschil kan zijn het soort gegevens, bijvoorbeeld de ene centrale uitgangssituatie met allerlei variaties van klanten, de andere met allerlei variaties van orders. Een andere mogelijkheid is een verschil in de tijd. Zo kan bijvoorbeeld een centrale uitgangssituatie vlak vóór de jaarovergang en vlak vóór de uitbetaling van vakantiegelden gedefinieerd worden, omdat dit belangrijke testmomenten zijn.

Daarnaast ontstaan bij het opstellen van de testspecificaties allerlei uitgangssituaties, normaal gesproken één per testscript. Hiervan overlegt de tester die de centrale uitgangssituatie beheert, met de tester van de script-uitgangssituatie welke gegevens geschikt zijn om toe te voegen aan de centrale uitgangssituatie. Hierbij kunnen bijvoorbeeld de volgende criteria gehanteerd worden:

- kunnen andere testers (een deel van) de script-uitgangssituatie hergebruiken?
- conflicteert de script-uitgangssituatie met (de consistentie van) de centrale uitgangssituatie?
- kan opname van de script-uitgangssituatie in de centrale uitgangssituatie andere tests verstoren?
- leidt opname van de script-uitgangssituatie in de centrale uitgangssituatie tot efficiëntievoordeel bij uitvoering van het script?

Voor het vullen van de centrale uitgangssituatie met testgegevens zijn er diverse mogelijkheden. Deze worden verderop beschreven.

De beschrijving van de centrale uitgangssituaties wordt conform de voor testware vastgelegde normen en standaards opgesteld en na afronding in beheer genomen.

Naamgeving testgegevens

Een aandachtspunt bij het zelf maken van de fysieke testgegevens is de naamgeving. Er kan voor gekozen worden om de gegevens net zo te noemen als in productie. Er worden dan natuurgetrouwe namen (maar wel fictief) gegeven aan bijvoorbeeld testpersonen, testadressen, testcodes, testgebruikers enzovoort.

Ook kan er voor gekozen worden om de gegevens een voor de test relevante naam te geven, bijvoorbeeld door het testgevalnummer, testeenheid, deelobject of testdoel in de naam op te nemen. Dit helpt ook bij het makkelijker oplossen van bevindingen en overdragen aan andere testers.

De derde optie is om betekenisloze namen te genereren. Voor het voorgaande voorbeeld van personen ontstaat dan:
Persoon1
Persoon2
Persoon3
Persoon4
enzovoort.

Deze laatste optie bespaart tijd in het uitzoeken en bedenken van natuurgetrouwe of testgerelateerde namen maar brengt ook een risico met zich mee. Het kan zijn dat bepaalde functionaliteit of een ander kenmerk van het systeem hierdoor

anders reageert. Voorbeelden zijn de werking van de sortering (die is nu vrij eenvoudig geworden en kan dus niet uitvoerig getest worden), lange persoonsnamen of letters met accenten. Een ander voorbeeld is de performance. Het databasemanagementsysteem kan op een tabel met 1000 fictieve namen die oplopend genummerd zijn anders behandelen dan een tabel met 1000 natuurgetrouwe namen. De zogenaamde index op een tabel kan anders worden opgebouwd wat niet ten goede komt aan de performance.

Het opbouwen van testgegevens

Voor het opbouwen van testgegevens kan gekozen worden uit drie alternatieven:

1. Opbouwen met reguliere systeemfuncties;
2. Opbouwen met aparte laadprogrammatuur;
3. Gebruik van productiegegevens.

1. Opbouwen met reguliere systeemfuncties

Opbouwen met reguliere systeemfuncties heeft als nadeel dat die functies zelf vaak nog niet uitputtend zijn getest en dat de ingevoerde gegevens daarom grondig gecontroleerd moeten worden. Het voordeel is dat tijdens het opbouwen van de bestanden de reguliere functies gelijktijdig impliciet worden getest en de consistentie tussen de gegevens gewaarborgd is. Een voorwaarde is wel dat de invoerfuncties als eerste worden opgeleverd. Dit moet tijdig worden afgesproken met de leverancier van de software.

2. Opbouwen met aparte laadprogrammatuur

Opbouwen met aparte laadprogrammatuur en testbestanden heeft als risico dat de testomgeving inconsistente of niet-toegestane situaties gaat bevatten, omdat er geen controle is op de invoer. Dit betekent dat bij de opbouw technische ondersteuning nodig is en er uiteraard (aan een test onderworpen) laadprogrammatuur beschikbaar moet zijn. Het voordeel is dat de bestanden relatief snel kunnen worden opgebouwd.

Uitgediept

Werken met 0-data, 0-scripts en 0-bestanden

0-Data zijn testgegevens die initieel in het systeem nodig zijn voor het uitvoeren van de test. 0-Data kunnen in vele vormen voorkomen. Zo kan het bestaan uit personen met een naam, adres, woonplaats en andere kenmerken die gebruikt worden in verschillende testgevallen. Ook kunnen het de gebruikers (users) zijn die het systeem mogen gebruiken (de testers). Weer een andere vorm is de data in zogenaamde stamtabellen. Het is van belang de benodigde 0-data te identificeren en te beschrijven bij de specificatie van de testgevallen.

0-Scripts zijn testscripts waarmee de 0-data in het systeem worden gebracht. Dit gebeurt via de reguliere systeemfuncties en dat heeft als voordeel dat betreffende functies van het systeem al getest worden. Bijkomend voordeel is dat exact duidelijk is wat de uitgangsituatie/data is (0-scripts worden uitgevoerd op een lege database). 0-Scripts worden als eerste uitgevoerd en dus kan de tester bij de uitvoering van 0scripts al een eerste idee krijgen van de kwaliteit van het testobject.

Voorwaarde voor het werken met 0-scripts is natuurlijk dat de functies die nodig zijn voor het invoeren van 0-data als eerste gebouwd worden. Wanneer dat niet het geval is kan er voor gekozen worden om te werken met zogenaamde **0-bestanden**. Deze bestanden bevatten de 0-data en via aparte laadprogrammatuur (bijvoorbeeld op basis van SQL) kunnen deze rechtsreeks in de database worden ingelezen.

3. Gebruik van productiegegevens

Het gebruik van productiegegevens als testgegevens heeft als voordeel dat met veel gegevens getest kan worden, dat de bestanden snel kunnen worden opgebouwd en dat impliciet de eventuele conversieprogrammatuur wordt getest.

Een nadeel is dat deze gegevens onderling weinig variatie vertonen en dat het veel uitzoekwerk kan betekenen om de juiste variatie in uitgangsgegevens bij een testgeval te zoeken. Een ander nadeel is dat het niet altijd is toegestaan

(wegens privacy-wetgeving of fraudegevoeligheid) met productiegegevens te werken. Hierdoor is het noodzakelijk om identificerende gegevens onherkenbaar te maken.

In sommige gevallen wordt niet een kopie van productie voor de test bevroren, maar wordt periodiek een nieuwe kopie in de testomgeving neergezet. Het nadeel hiervan is dat de tests niet zonder meer herhaalbaar zijn, omdat de productiegegevens elke kopie weer anders zijn, waardoor de testresultaatvoorspellingen niet meer kloppen.

Tip

Een variant op het verkrijgen van testgegevens uit productie is het laten aanleveren van testgegevens door gebruikers. Zij kennen als geen ander het systeem met de bijbehorende gegevens en dus als geen ander de 'moeilijke' gevallen. Vraag aan elke gebruiker een aantal moeilijke gevallen in de vorm van testgegevens. Dit kan door de gebruiker zelf achter het testobject plaats te laten nemen en deze gevallen te laten invoeren. Een andere mogelijkheid is om de specifieke gevallen uit productie te kopieren en in de testomgeving neer te zetten.

Los van planningstechnische en budgettaire perikelen, heeft het eerste alternatief, opbouw met reguliere systeemfuncties, de voorkeur. Indien het testteam toestemming heeft om vanuit productie testbestanden te halen, is het ook mogelijk om de drie alternatieven te combineren. Kies een zodanige verzameling van productiegegevens dat bijvoorbeeld voor elk soort gegeven (klant, order, factuur, enzovoort) een bepaalde vulling aanwezig is. Deze subset wordt geladen in de testomgeving (met behoud van de consistentie tussen de verschillende gegevens). Vervolgens worden met behulp van reguliere systeemfuncties wijzigingen aangebracht in deze gegevens, om zo de gewenste uitgangssituatie te creëren.

Uitgediept

Testgegevens bij het testen van datawarehouses

Een datawarehouse valt op hoofdlijnen op te splitsen in twee groepen programma's:

- de extractie- en conversieprogramma's om de datawarehouse te vullen;
- de rapportageprogramma's om de informatie uit de datawarehouse te halen.

Hoewel voor het testen van individuele extractie-en conversieprogramma's het gebruik van aparte testgegevens de voorkeur heeft, wordt bij het integraal testen van de rapportageprogramma's al snel gebruik gemaakt van productiegegevens. De reden hiervoor is dat het opstellen van een consistente set van fictieve testgegevens veel inspanning kost en dat bij een set productiedata deze consistentie bijna automatisch gewaarborgd is. Bovendien is een groot voordeel dat een gebruiker de uitkomst van een rapport bij gebruik van echte productiegegevens makkelijker kan beoordelen.

Nadelen van het gebruik van productiedata bij het testen van een datawarehouse zijn echter:

- de moeilijkheid van het doen van precieze uitvoervoorspellingen, omdat moeilijk te achterhalen is wat de invoer is;
- de confidentialiteit (privacy-gevoeligheid) die met sommige gegevens gepaard gaat. In de praktijk betekent dit dat het gebruik van productiegegevens niet, of pas na toepassing van zogenaamde scrambling technieken (depersonaliseren, onherkenbaar maken van gegevens) mogelijk is;
- de continu veranderende situatie: de productiedata van vandaag zijn anders dan de productiedata van een week geleden, wat hertesten erg bemoeilijkt.

Dit laatste nadeel kan worden ondervangen door het dagelijks/wekelijks opnieuw laden van data tijdelijk stop te zetten, zodat men wel steeds van dezelfde uitgangssituatie gebruik kan maken. Een toegepaste versimpeling is dat niet het totale productiebestand wordt genomen, maar een selectie hieruit. Dit vraagt dan wel aandacht (en tijd) voor onderlinge consistentie van de gegevens.

Delta-test1

Als toevoeging hierop kan de volgende procedure doorlopen worden:

- Neem een subset van productiedata en noem deze X.
- Laat deze subset X in zijn geheel door de datawarehouse lopen en leg de resultaten vast.

- Vul nu de subset X aan met een aantal zelf bedachte testgevallen en noem deze set X+1.
- Laat deze subset X+1 ook in zijn geheel door de datawarehouse lopen.
- De resultaten van X+1 kunnen voorspeld worden door de resultaten van X aan te vullen met de zelf bedachte testgevallen.
- Vul op zijn beurt subset X+1 ook weer aan met testgevallen en noem deze X+2.
- Laat deze subset X+2 ook in zijn geheel door de datawarehouse lopen.
- De resultaten van X+2 kunnen voorspeld worden door de resultaten van X+1 aan te vullen met de zelf bedachte testgevallen (deze tweede doorloop is handig om veranderingen in de tijd te controleren).

Delta-test2

Een iets eenvoudiger variant op bovenstaande is nog de volgende;:

- Maak de database(s) van de aanleverende systemen leeg.
- Zet een aantal zelf gespecificeerde testgevallen in deze systemen.
- Draai de extractie en controleer het resultaat in de datawarehouse.
- Zet als een soort regressietest dezelfde testgevallen nu in de volle database(s) van de aanleverende systemen.
- Draai de extractie en controleer het resultaat (van de testgevallen) in de dataware house.

Voorbeeld

Bij een testtraject van een grote datawarehouse worden als testgegevens de volgende testbestanden gebruikt:

- De kleine testset: dit is een zo klein mogelijk testbestand, waarbij op doelmatige wijze de mogelijk voor de hand liggende functionele problemen bij het gebruik van het prototype worden opgezocht. Deze testset wordt gebruikt als eerste test na het ontwikkelen of herstellen van het prototype en bij alle andere tests om snel een beeld te krijgen.
- Het 1000-records testbestand wordt gebruikt voor de functionele acceptatietest en bestaat uit plusminus 1000 records van een dagbestand. Het dagbestand dat hiervoor gebruikt wordt moet een dag zijn waarbij zoveel mogelijk (probleemgenererende) verschillende gevallen voorkomen. De keuze hiervan wordt samen met de klant bepaald.
- De 5% (of X%) testset is een door de klant samengestelde representatieve steekproef uit de bronbestanden voor het derde increment.
- De testset ‘dagbestanden’ bestaat uit een compleet dagbestand. Het dagbestand dat hiervoor gebruikt wordt moet een dag zijn waarbij zoveel mogelijk (probleemgenererende) verschillende gevallen voorkomen. De keuze hiervan wordt samen met de klant bepaald. Het uitvoeren van een weekproces om zo te controleren of beginstanden + mutaties = eindstanden is hierbij een belangrijk aandachtspunt. Voorlopig wordt uitgegaan van de dagen woensdag 1 maart, donderdag 2 maart en vrijdag 3 maart, waarna een weekproces wordt gedraaid om dit te controleren.

Gebruik van uitgangssituaties gedurende de test

Over het gebruik van de centrale uitgangssituatie gedurende de test moet vooraf worden nagedacht. Hierbij gaat het vooral om de keuze tussen:

1. het cumulatief opbouwen van de centrale uitgangssituatie (ongestructureerd en gestructureerd);
2. het periodiek verversen met de centrale uitgangssituatie (master-copy);
3. het parallel gebruiken van meerdere versies.

1. Het cumulatief opbouwen van de centrale uitgangssituatie (ongestructureerd en gestructureerd)

Bij cumulatieve opbouw groeit de centrale uitgangssituatie met de tests mee. Wanneer dit ongestructureerd gebeurt, voeren de testers naar behoefte nieuwe testgegevens in. Dit geeft de testers veel vrijheid en flexibiliteit, maar heeft ook een nadeel. Verschillende testers voeren hun eigen testgegevens in die de testresultaten van een andere test kunnen beïnvloeden. Bij analyse van testresultaten kan hierdoor veel tijd verloren gaan met uitzoeken. Bovendien zullen gegevens snel onderling inconsistent worden.

Bij de gestructureerde variant maken de testers afspraken om bovenstaande beinvloeding te voorkomen. Zo kunnen ze afspreken dat alleen bepaalde soorten testgegevens ingevoerd of gewijzigd mogen worden of dat testgegevens een identificatie moeten hebben waardoor herkenbaar is van welke tester deze zijn.

Voorbeeld

Bij de test van een systeem voor de afrekening van mobiele telefoonabonnementen (ook vaak een billing-systeem genoemd) was een testteam betrokken van 5 personen. Ieder van deze personen was verantwoordelijk voor de test van een specifiek deelsysteem. Om te voorkomen dat de testers elkaar in de weg zouden zitten bij het gebruik van de uitgangssituatie, werd voorgesteld om aan ieder deelsysteem een range van telefoonnummers te koppelen. De uitgangssituatie van de testgevallen voor een specifiek deelsysteem moest dan binnen die range vallen. Daarnaast werd een range afgesproken voor de integrale test die over de verschillende deelsystemen liep. Dit leverde de volgende verdeling op:

Deelsysteem 1: range telefoonnummers +31610000000 tot +31619999999
Deelsysteem 2: range telefoonnummers +31620000000 tot +31629999999
Deelsysteem 3: range telefoonnummers +31630000000 tot +31639999999
Deelsysteem 4: range telefoonnummers +31640000000 tot +31649999999
Deelsysteem 5: range telefoonnummers +31650000000 tot +31659999999
Integraal: range telefoonnummers +31690000000 tot +31699999999

2. Het periodiek verversen met de centrale uitgangssituatie (master-copy)

Een tweede aanpak is het regelmatig terugzetten van de centrale uitgangssituatie (ook wel de “master-copy” genoemd). Dit wordt gedaan via een back-up- en restoreprocedure. Er wordt als eerste een back-up gemaakt van de master. Op bepaalde momenten zet de beheerder van de master deze weer terug. Dat kan periodiek zijn, bijvoorbeeld elke dag of week, maar ook op aanvraag, bijvoorbeeld na het uitvoeren van een test. Een speciale beheerprocedure kan voorzien in het structureel toevoegen van testgegevens aan de master. Een groot voordeel is de beheersbaarheid van de gegevens, maar nadelen zijn de afhankelijkheid van het verversingsmoment en het extra werk om vanuit de master tot de voor de test benodigde uitgangssituatie te komen.

3. Het parallel gebruiken van meerdere versies

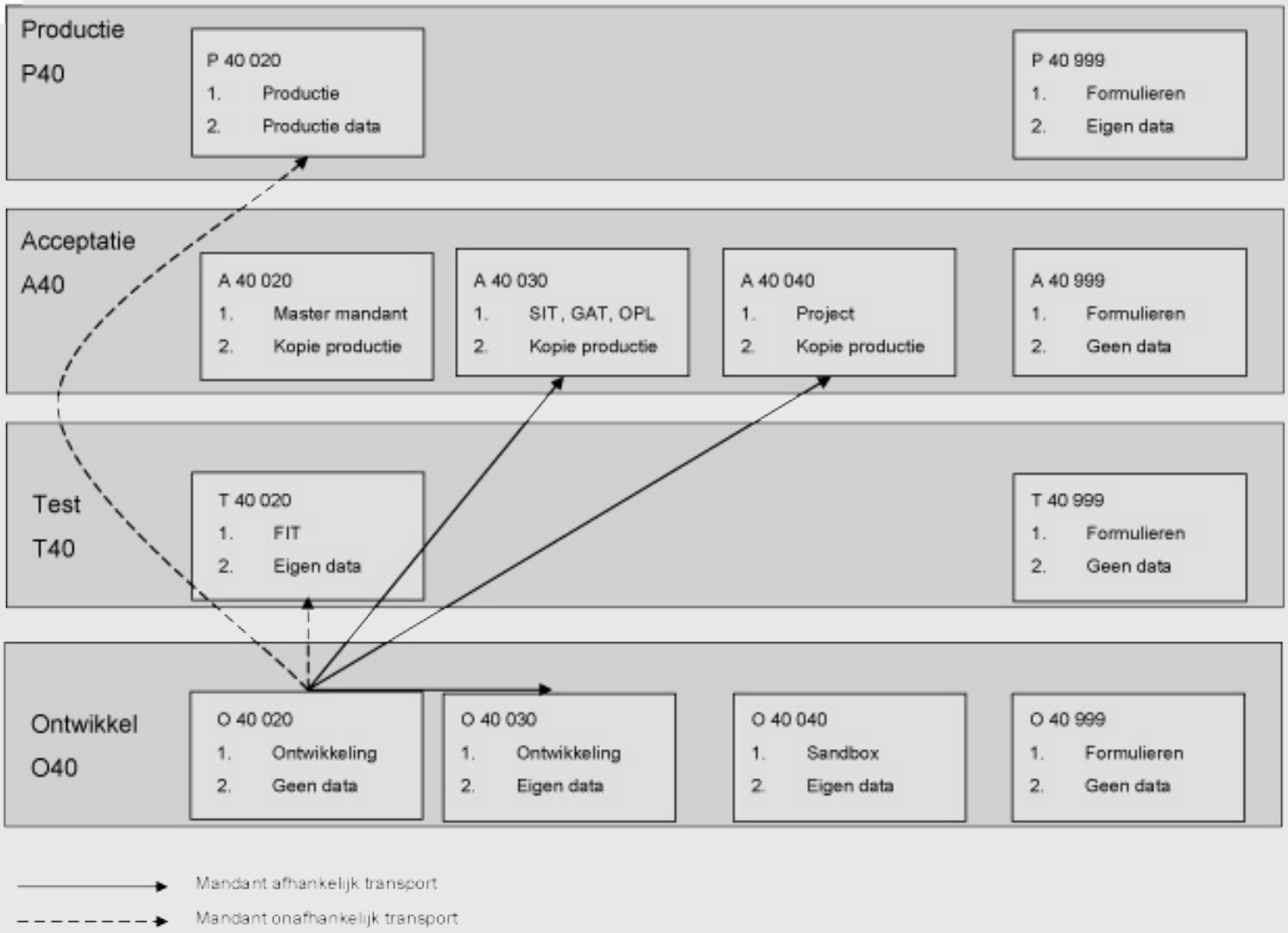
Een derde mogelijkheid is het gebruik van meerdere omgevingen met parallelle versies van de gegevens. Elke tester heeft zijn eigen testomgeving en uitgangssituatie(s). Het beschikken over een centrale uitgangssituatie kan handig blijven, maar elke tester kan deze naar believen in eigen omgeving aanpassen. Grote voordeel van deze aanpak is de onafhankelijkheid van de tests: verstoring door andere tests is nauwelijks mogelijk, de tester weet precies wat er in de eigen uitgangssituatie zit. Dat levert enorme tijdwinst op. Een nadeel is dat door het isolement van de tests fouten in uitgangssituaties heel lang onontdekt kunnen blijven en integrale testaspecten pas laat aan de orde komen. Een ander nadeel zijn de extra kosten voor de benodigde testomgevingen, zowel in termen van hardware als beheer.

Een belangrijke voorwaarde voor deze werkwijze is goed configuratiebeheer. Dit moet zorgen dat op alle testomgeving de opleveringen van software en de naleveringen naar aanleiding van opgeloste bevindingen synchroon worden uitgerold. Dit kan dus een risicofactor zijn.

Uitgediept

Testomgevingen en testgegevens binnen SAP®

In de terminologie van SAP spreekt men van een systeemlandschap waarin de verschillende omgevingen zitten. Een systeemlandschap bestaat vaak uit een aparte ontwikkel-, test-, acceptatie- en productieomgeving (ook wel OTAP geheten). Deze omgevingen noemt men mandanten of clients. Er kunnen meerdere mandanten per omgeving (instances) aanwezig zijn. Meerdere mandanten zorgen ervoor dat de testers wat testgegevens betreft elkaar niet in de weg zitten. Het is aan te raden een aparte master-mandant aan te leggen om de testgegevens veilig te stellen. Door kopiëren kan deze data naar een andere mandant worden gezet. Tevens heeft SAP de tool Test Data Migration Server, waarmee data van een productieve omgeving gereduceerd en eventueel geanonimiseerd overgezet kan worden naar niet productieve omgevingen.



Figuur 6.18: SAP-omgevingen en transporten.

In figuur 6.18 staat een schematisch voorbeeld van de omgevingen en bijbehorende transporten.

Het overzetten van wijzigingen (customizing, nieuwe programmatuur) van SAP van de ene omgeving naar de andere gebeurt door middel van zogenaamde transporten naleveringen naar aanleiding van opgeloste bevindingen synchroon worden uitgerold. Dit kan dus een risicofactor zijn.

(SAP Transport Management Systeem). Transporten kunnen mandant-afhankelijk of mandant-onafhankelijk zijn. Bij transporten is het noodzakelijk dat een bepaalde volgorde gehanteerd wordt en soms is het nog nodig dat per omgeving handmatig bepaalde instellingen worden gezet. Dit alles vraagt om een zeer gedegen configuratiemanagement met daarin release- of transportadministratie.

Testgegevens bij uitbesteding

Een ontwikkeling die in de belangstelling staat van verschillende wet- en regelgevers, is de omgang met elektronische gegevens bij uitbesteding. Twee onderwerpen verdienen hierbij speciale aandacht:

- Vertrouwelijkheid van de gebruikte gegevens
  Steeds vaker wordt in wetten of formele richtlijnen vastgelegd hoe om te gaan met digitale persoonsgegevens en hoe te garanderen dat deze informatie vertrouwelijk blijft. Wanneer testgegevens worden gecreëerd uit productiedatabases is het bij uitbesteding noodzakelijk dat de data anoniem wordt gemaakt. De data verlaat namelijk de organisatie en steeds vaker ook het land. Zo zijn al gevallen bekend waarin werknemers van de leverancier misbruik hebben gemaakt van de software en de gegevens van de uitbestedende organisatie.

Tip

Let er bij het anonimiseren op dat alle gegevens op dezelfde wijze worden geanonimiseerd zodat de gegevens in diverse bestanden onderling consistent blijven.

- Verantwoordelijkheid voor aanlevering gegevens
  Een ander aandachtspunt is het specificeren van de testgevallen en de benodigde 0-data. De leverancier heeft soms onvoldoende materiekennis om zelf enigszins reële waarden te bedenken. Extreme voorbeelden: postcodetabellen met 3 in plaats van 4 numerieke posities gebruiken of het BTWpercentage op 100% zetten. Dit kan het uitvoeren van tests erg verstoren en maakt bovendien een controle op de testresultaten bijzonder lastig. Als bepaalde 0-data belangrijk zijn voor een goede test, dan moeten er afspraken gemaakt worden wie wanneer deze testgegevens aanlevert.

Producten

Testbasisbevindingen.
Een beschrijving van de centrale uitgangssituatie(s).

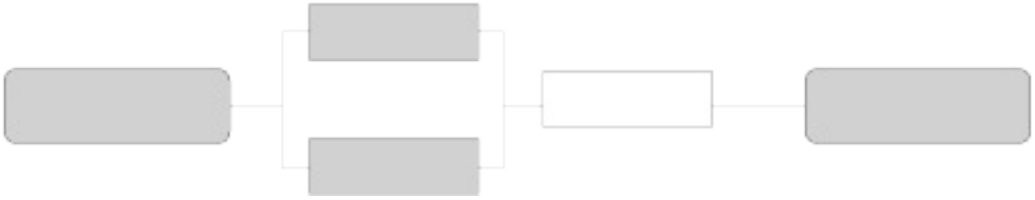
Technieken

Niet van toepassing.

Tools

Testdatatool.

6.6.3 Specificeren intake testobject



Doel

Het voorbereiden van de intake van het testobject zodat deze zo snel mogelijk kan starten na oplevering van het testobject.

Werkwijze

Deze activiteit omvat de volgende deelactiviteiten:

1. Opstellen checklist intake testobject;
2. Opstellen testscript pretest.

1. Opstellen checklist intake testobject

Op enig moment ontvangt het testteam het testobject. Deze eerste activiteit heeft tot doel om vast te stellen of de oplevering van het testobject volledig is, dat wil zeggen datgene bevat wat afgesproken is met de leverancier van het testobject, niet minder maar ook niet meer dan dat. Zo bestaat het testobject normaal gesproken uit allerlei softwarecomponenten (met elk een bepaalde versie), maar ook gebruikershandleiding en installatiehandleiding kunnen bijvoorbeeld deel van het testobject zijn. De tester legt in de checklist vast welke onderdelen worden verwacht bij de oplevering.

Naast informatie over het testobject zelf bevat de checklist ook vragen over de informatie van de oplevering. Er moet zichtbaar zijn welke wijzigingen de oplevering bevat en welke onderdelen aan welke wijziging gerelateerd zijn. Zo kan

voorkomen worden dat het testteam gewijzigde softwareonderdelen ontvangt, terwijl het testteam daarvoor helemaal geen wijzigingsvoorstel heeft en er dus ook geen test voor gepland heeft.

Uitgediept

Met name bij specialistische pakketten doet dit verschijnsel zich voor, omdat de leverancier de wijzigingsvoorstellen van een aantal klanten tegelijk implementeert, maar naar elke klant afzonderlijk alleen de door deze gevraagde wijzigingen terugkoppelt.

2. Opstellen testscript pretest

Na installatie van het testobject vindt een pretest plaats om te bepalen of het testobject goed genoeg is om te gaan testen. In deze activiteit bereiden de testers deze pretest voor door een testscript op te stellen. Dit kan in meerdere gradaties van zwaarte. Hieronder staan enkele voorbeelden:

- checklist met alle functies, deze moeten allemaal benaderbaar zijn;
- voor een aantal representatieve functies wordt een eenvoudig testgeval met valide invoer (“goed-geval”) gespecificeerd;
- specificatie van enkele op integratie gerichte testgevallen om te controleren dat de verschillende onderdelen met elkaar kunnen communiceren.

De gegevenscyclustest is hiervoor een goede keuze (paragraaf 14.4.7).

De testgevallen mogen uit de reguliere tests gehaald worden, maar de resultaatcontrole is veel soepeler. Zo is het voor de pretest niet belangrijk dat het testgeval een correct resultaat levert, als het maar een resultaat levert en bijvoorbeeld niet crasht.

Voorbeelden

- Voor een bank-applicatie bestaat de pretest uit een script van 25 end-to-end testgevallen.
- Bij een andere financiële organisatie duurt de pretest een dag waarin een tester de testgevallen uitvoert, die de belangrijkste functionaliteit bevatten.
- Een telecommunicatie-organisatie laat als pretest een aantal end-to-end testgevallen door de leverancier van de software uitvoeren.

Producten

Checklist intake testobject.
Testscript pretest.

Technieken

Niet van toepassing.

Tools

Niet van toepassing.

6.7 Fase Uitvoering

Doel

Het verkrijgen van inzicht in de kwaliteit van het testobject door het uitvoeren van de afgesproken tests.

Context

De feitelijke test vindt in deze fase plaats. Het testobject, of een afzonderlijk testbaar deel van het testobject, is opgeleverd en in de voorgaande fasen is zoveel mogelijk voorbereid om de testuitvoering zo kort mogelijk te houden.

Randvoorwaarden

Aan de volgende voorwaarden dient te zijn voldaan alvorens kan worden gestart met de fase Uitvoering:

- het testobject, of een afzonderlijk testbaar deel van het testobject, moet opgeleverd zijn;
- de testscripts voor het testobject, of afzonderlijk testbaar deel van het testobject, moeten beschikbaar zijn;
- van de bijbehorende testinfrastructuur moet de intake succesvol zijn verlopen.

Werkwijze

De daadwerkelijke uitvoering van de test begint op het moment dat het testobject, of een afzonderlijk testbaar deel van het testobject, wordt opgeleverd. Eerst wordt het testobject gecontroleerd op volledigheid. Hierna wordt het in de testomgeving geïnstalleerd om te kunnen beoordelen of het geheel naar behoren functioneert. Dit gebeurt door het uitvoeren van een eerste test, de zogenaamde pretest. In deze test wordt globaal getest, met als doel te onderzoeken of het te testen informatiesysteem, in samenhang met de testinfrastructuur, van voldoende kwaliteit is om uitgebreid getest te kunnen worden. Als het geheel van voldoende kwaliteit is, wordt de centrale uitgangssituatie klaargezet. De test kan worden uitgevoerd op basis van de (handmatige of geautomatiseerde) testscripts die in de fase Specificatie zijn opgesteld. In dat geval moeten eerst de uitgangssituaties voor de uit te voeren testscripts worden klaargezet. De uitvoering van de test kan ook op een explorerende manier worden uitgevoerd of op basis van checklists. Tijdens de uitvoering worden de testresultaten bijgehouden. Het onderzoek naar wat de oorzaak is van de eventuele verschillen tussen de verwachtingen en de verkregen testresultaten, vindt na de uitvoering plaats. Oorzaak voor verschillen kan gelegen zijn in een softwarefout, maar er zijn ook andere oorzaken mogelijk. Zo kunnen er fouten zitten in de testbasis, in de testomgeving of in de testgevallen. Wanneer een fout opgelost moet worden, dan wordt deze formeel aangemeld als bevinding. Wanneer deze bevinding is opgelost kan een nieuwe test worden uitgevoerd. Zo ontstaat tijdens deze fase vaak een iteratief proces van test-herstel-hertest. De invulling van dit iteratieve proces is afhankelijk van de oorzaak van de fout. Zo kan een fout in de testbasis resulteren in het hernieuwd (her)plannen van de test waarna de fasen Voorbereiding, Specificatie en Uitvoering weer worden doorlopen. Bij een fout in de software kan het iteratieve proces van test-herstel-hertest beperkt blijven tot het weer uitvoeren van de fase Uitvoering.

Rollen / verantwoordelijkheden

Alle activiteiten kunnen worden uitgevoerd door alle testteamleden. Alleen de controle op volledigheid van het testobject wordt gedaan door de testmanager bijgestaan door (wanneer deze rol is ingevuld) de testinfrastructuurcoordinator.

Activiteiten

Binnen de fase Uitvoering worden de volgende activiteiten onderkend:

1. Intake testobject;
2. Klaarzetten uitgangssituatie;
3. Uitvoeren (her)tests;
4. Controleren en beoordelen testresultaten.

Het volgende schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten;

Figuur 6.19: Uitvoering.

6.7.1 Intake testobject

Doel

Het vaststellen of de opgeleverde delen van het testobject zodanig functioneren dat er zinvol getest kan worden.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Controle volledigheid opgeleverd testobject;
2. Uitvoeren pretest.

1. Controle volledigheid opgeleverd testobject

Met behulp van de in de fase Specificatie opgestelde checklist wordt de volledigheid van het opgeleverde testobject gecontroleerd. Dit wordt gedaan door de testmanager, bijgestaan door (wanneer deze rol is ingevuld) de testinfrastructuurcoördinator. Ontbrekende delen worden, door middel van een bevinding, gerapporteerd aan de betrokken partijen. Wanneer de bevinding testblokkerend is (dus de volgende deelactiviteit pretest kan niet starten), dan moet deze meteen worden opgelost. Het is aan te raden deze deelactiviteit samen uit te voeren met de afdeling die de testomgeving beheert. Deze afdeling is namelijk afhankelijk van een compleet opgeleverd testobject omdat anders de installatie fout loopt. Bovendien hebben zij de technische kennis in huis om het testobject op het aspect van volledigheid te controleren. Na een akkoord kan de testinfrastructuurcoördinator de installatie van het testobject (door de beheerders laten) uitvoeren.

2. Uitvoeren pretest

Zodra een (versie van het) testobject is geïnstalleerd, is het van belang om een zogenoemde pretest uit te voeren. Deze pretest vindt plaats voordat het werkelijke testen start. De bedoeling van de pretest is om te beoordelen of het testobject van voldoende kwaliteit is om te gaan testen. De pretest wordt uitgevoerd met het testscript dat hiervoor is opgesteld in de fase Specificatie (zie paragraaf 6.6.3 “Specificeren intake testobject”). Het komt in de praktijk regelmatig voor dat systemen de eerste testdag(en) niet juist worden opgeleverd of geïnstalleerd, waardoor er niet gestart kan worden met de uitvoering van de test. Dit is niet alleen zonde van de tijd; het bevordert evenmin de motivatie van het testteam. Het is belangrijk om bij het opstellen van de planning in het testplan hier al rekening mee te houden dat dit tijd kost (zie ook paragraaf 6.2.7. “Bepalen planning”).

Tip

De pretest als onderhandelingspositie

De testmanager staat met de pretest veel sterker om te zeggen dat de testtijd nog niet is gaan lopen. Dus wanneer vanaf maandag 10 dagen testuitvoering gepland stonden, en de pretest pas woensdag slaagt, gaan de 10 dagen pas in vanaf woensdag. Hier kan dan nog best over gediscussieerd worden, maar de testmanager heeft een veel sterkere onderhandelingspositie.

Voorwaarde voor het uitvoeren van deze deelactiviteit is dat de benodigde testinfrastructuur beschikbaar is. Dit betekent concreet dat de intake van de infrastructuur succesvol moet zijn verlopen (zie paragraaf 6.4.4 “Intake infrastructuur”). Een goed verloop van de pretest is voorwaarde voor het starten van de volgende activiteiten in de fase Uitvoering. De bevindingen uit de pretest worden geregistreerd en indien een bevinding testblokkerend is, wordt deze onmiddellijk beschikbaar gesteld aan de betrokken partijen. Alles moet, met de hoogste prioriteit, in het werk worden gesteld om deze testblokkerende bevindingen op te lossen en de pretest alsnog succesvol te voltooien.

Voorbeeld 1

Bij de bouw en test van een nieuw registratiesysteem ontstaat uitloop en er wordt voor gekozen om in het weekend te gaan testen. Dit betekent dat de testers gevraagd wordt hun vrije zaterdag en zondag op te offeren. Ervaringen uit het verleden hebben geleerd dat de oplevering en installatie van het testobject in de testomgeving niet altijd probleemloos verloopt. In veel gevallen ontbreken er delen van het testobject of werkt het in het geheel niet.

Om nu te voorkomen dat de testers op zaterdag voor niks komen opdagen omdat de installatie van het testobject niet is gelukt wordt besloten op vrijdagavond een pretest uit te voeren. Deze wordt uitgevoerd door de testmanager en testinfrastructuurcoördinator. Samen controleren ze om 18:00 uur op basis van een selectie van de testscripts de kwaliteit van het opgeleverde testobject. Wanneer deze voldoende is om getest te worden, nemen ze voor 20:00 uur contact op met de testteamleden om deze te vertellen dat de test doorgaat.

Voorbeeld 2

Een nieuwe versie van een administratief systeem wordt ontwikkeld door een externe leverancier aan de andere kant van de wereld. Met de leverancier is afgesproken dat, voordat de versie wordt opgeleverd en de FAT wordt uitgevoerd, eerst een pretest plaatsvindt. Deze test wordt, via internet, uitgevoerd op de infrastructuur van de leverancier. Zo kan een eerste indruk van de kwaliteit van het systeem worden verkregen en vertrouwen ontstaan dat de FAT werkelijk kan beginnen. Ook wordt zo voorkomen dat eventuele problemen met de installatie van het testobject in de eigen infrastructuur zorgt voor wantrouwen richting de leverancier. Daar hebben ze het immers zien werken!

Producten

Bevindingen.
Geïnstalleerd en testbaar testobject.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool.
Bevindingenbeheertool.
Geautomatiseerde testuitvoering tool.
Monitor.
Comparator.
Databasemanipulatietool.
Simulator.
Stubs en drivers.

6.7.2 Klaarzetten uitgangssituatie

Doel

Het klaarzetten van de uitgangssituatie die benodigd is voor de uitvoering van de tests.

Werkwijze

Voordat gestart kan worden met de uitvoering van de testgevallen in het testscript, moet het testobject in de benodigde toestand of situatie gezet worden. Hieronder valt niet alleen het klaarzetten van de testgegevens die voor de 'processing' nodig zijn. Ook het zetten van het systeem en testomgeving in een bepaalde toestand hoort hier bij. Dit kan bijvoorbeeld het formatteren van een disk zijn, maar ook het configureren van een invoerapparaat.

Er worden twee soorten situaties van het testobject onderscheiden binnen TMap:

1. een *centrale uitgangssituatie* voor een aantal tests;
2. een *uitgangssituatie* per testscript.

Bij aanvang van een test wordt de centrale uitgangsituatie klaargezet. Het testobject en de testomgeving zijn dan klaar om de input volgens alle testscripts (of in ieder geval degene die als eerst worden uitgevoerd) te ontvangen. De testgegevens worden opgebouwd zoals beschreven tijdens de activiteit "Definiëren centrale uitgangssituatie(s)" in de fase Specificatie (zie paragraaf 6.6.2). Het opbouwen van deze testgegevens kan op verschillende manieren plaatsvinden. Bevindingen tijdens de opbouw van de testgegevens worden geregistreerd conform de in het testplan vastgestelde procedures.

Tip

Back-up van de centrale uitgangsituatie en controle daarvan

Zodra het testobject in de centrale uitgangsituatie is gezet en gecontroleerd, is het aan te bevelen een back-up te maken. Deze kan op ieder gewenst moment weer worden teruggezet. Het is belangrijk om dit principe van back-up en terugzetten goed te controleren voor aanvang van de tests.

Producten

Bevindingen.
(Centrale) Uitgangssituatie.
Initiële situatie.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool.
Testdatatool.
'Model based testing'-tool.
Geautomatiseerde testuitvoering tool.
Performance-, load-, en stresstesttool.
Databasemanipulatietool.

6.7.3 Uitvoeren (her)tests

Doel

Het verkrijgen van testresultaten op basis waarvan de evaluatie van het testobject kan plaatsvinden.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Uitvoeren dynamische expliciete tests;
2. Uitvoeren dynamische impliciete tests;
3. Uitvoeren statische tests.

1. Uitvoeren dynamische expliciete tests

Bij dynamisch expliciet testen worden expliciete testgevallen uitgevoerd om informatie over het te testen kenmerk (kwaliteitsattribuut) of systeemonderdeel te verkrijgen. Door het runnen van software en het uitvoeren van handelingen op het testobject worden de testresultaten verkregen. Deze resultaten worden in de volgende activiteit vergeleken met het verwachte resultaat waardoor eventueel bevindingen ontstaan. Dynamisch expliciet testen is de meest gangbare manier van testen. Er zijn 2 mogelijke vormen van dynamisch expliciet testen:

- Testen op basis van gespecificeerde tests opgesteld in de fase Specificatie.
  De gespecificeerde tests die zijn opgesteld in de fase Specificatie vormen het uitgangspunt voor de hier uit te voeren testen. Dit kunnen testscripts zijn die de test acties en controles van de fysieke testgevallen bevatten. De testscripts zijn beschreven in optimale volgorde en vormen het stappenplan voor de testuitvoering. Wanneer er voor gekozen is om gebruik te maken van tools voor geautomatiseerde testuitvoering dan worden de gespecificeerde tests met behulp van een testtool uitgevoerd (zie ook paragraaf 8.5 “Testtools”). Daarnaast kunnen ook tests plaatsvinden op basis van checklists of een andere vorm zoals beschreven in paragraaf 10.3 “Testvormen”. Belangrijke voorwaarde voor een waardevolle dynamische expliciete test is dat de testers niet afwijken van de testgevallen en minimaal de beschreven testgevallen uitvoeren. Anders is op geen enkele wijze te garanderen dat de in het testplan vastgestelde strategie daadwerkelijk wordt uitgevoerd.

- Testen op basis van een explorerende techniek.
  In deze vorm gaat de tester explorerend te werk tijdens de dynamische expliciete test. Dit betekent dat de tester steeds een stukje van de te testen applicatie verkent (exploreert), nadenkt over wat getest moet of kan worden (testontwerp) en dit vervolgens ook doet (testuitvoering). Hierbij doet de tester gelijk al weer nieuwe kennis op over de applicatie, denkt na over wat vervolgens getest moet worden, voert dit uit, enzovoort. Het ontwerpen en vervolgens uitvoeren van de tests gebeurt daarmee zeer dicht op elkaar. Mogelijke technieken hiervoor zijn “ Exploratory Testing” en “ Error Guessing”. Deze worden toegelicht in paragraaf 14.4 “Een basisset testontwerptechnieken”.

Tip

Een snelle manier om onervaren testers op gang te krijgen is om deze activiteit in paren uit te voeren. Dus een onervaren tester samen te laten testen met een ervaren tester [Kaner, 2001]. Hierbij is één tester verantwoordelijk voor de test. Deze tester schakelt een andere tester in, waarbij één de toetsen bedient en de ander de te testen zaken bedenkt, oplet, aantekeningen maakt en dingen uitzoekt. Door hardop na te denken genereren de testers gezamenlijk veel meer ideeën dan ieder afzonderlijk. Ook helpen ze elkaar om het doel van de test niet uit het oog te verliezen door onbelangrijke details. Coaching van paren is, zeker in het begin, aan te bevelen. Testen in paren werkt minder goed bij heel introverte of juist heel overheersende personen.

Deze twee vormen van dynamisch expliciet testen sluiten elkaar niet uit. Juist wanneer gecombineerd toegepast, kunnen ze elkaar versterken. Redenen voor het combineren van deze twee vormen kunnen zijn:

- Tijdens de uitvoering van de gespecificeerde tests is het gevoel ontstaan dat deze niet voldoende inzicht in de kwaliteit geven. Door nu aanvullend op een aantal plekken in het testobject explorerend te testen kan dit gevoel worden bevestigd of juist niet.
- De strategie voor een hertest kan zijn dat alleen die delen van het testobject worden getest die door de programmeurs zijn aangepast. Om toch zeker te zijn dat de ongewijzigde delen nog steeds werken kunnen deze extra op explorerende wijze worden getest.
- De aanvulling van explorerend testen boven op het testen met gespecificeerde tests kan nuttig zijn om het creatief testen te stimuleren. Dit kan dan bijvoorbeeld ingepland worden op de vrijdagmiddag. Dit dagdeel van de week is kenmerkend voor de mate van creativiteit waarin veel testers dan verkeren. Zo met het weekend voor de deur heerst er een opperbeste stemming, waar iedereen open staat voor nieuwe experimenten. Deze experimenten kunnen hele uitzonderlijke situaties bevatten maar ook juist situaties die zo gewoon zijn dat ze vergeten worden. Juist dan kunnen zeer cruciale fouten in het testobject worden gevonden.

- Tijdens de uitvoering van een testscript kan een bevinding optreden. Deze bevinding moet, voordat deze gerapporteerd wordt, verder onderzocht worden. Zo kan er gekeken worden of de bevinding zich altijd voordoet of juist alleen in de specifieke situatie. Of misschien komt de bevinding op meer (gelijksoortige) plekken voor in het testobject. Ook bestaat de mogelijkheid dat er meerdere bevindingen bij elkaar zitten. Dit onderzoek kan op basis van explorerend testen plaatsvinden (zie hoofdstuk 12 “Bevindingenbeheer”).

Tip

Fouten bij elkaar

Fouten in een testobject hebben de neiging om bij elkaar te zitten. Wanneer in een bepaalde functie, scherm, afhandeling of ander onderdeel een fout zit, dan is de kans groot dat daar ook nog andere fouten zitten. Hiervoor zijn verschillende oorzaken aan te dragen. Zo kan juist dat onderdeel zeer complexe code bevatten en dus is de kans dat de programmeur een fout maakt groter. Ook kan juist een bepaald deel door een onervaren programmeur gemaakt zijn of door een programmeur die op dat moment niet lekker in zijn vel zat. Het is daarom aan te raden om bij het constateren van een fout altijd op zoek te gaan naar meerdere fouten in de buurt.

2. Uitvoeren dynamische impliciete tests

Tijdens het dynamisch expliciet testen kan ook informatie over andere kenmerken (kwaliteitsattributen) worden verzameld. Hiervoor zijn geen expliciete testgevallen ontworpen. Dit wordt dynamisch impliciet testen genoemd. Deze tests kunnen gepland en ongepland worden uitgevoerd. Wanneer het gepland gebeurt, wordt vóóraf afgesproken dat dit een wezenlijk onderdeel vormt van de teststrategie. Testers kunnen dan vooraf aan de testuitvoering gevraagd worden op een aantal kenmerken (zoals bijvoorbeeld de performance of de usability) van het testobject te letten. Hier liggen dan geen gerichte testgevallen aan ten grondslag. Een andere manier is om het de testers achteraf, na de uitvoering van de dynamisch expliciete test, te vragen. Gevaar is dan wel dat, doordat er niet wezenlijk aandacht aan is besteed, verkeerde informatie wordt gegeven.

Ongeplande impliciete tests ontstaan doordat er tijdens de uitvoering van de test bepaalde dingen beginnen op te vallen. Er wordt dan afgesproken hier meer op te gaan letten. Als bijvoorbeeld regelmatig systeemstoringen optreden, kan een uitspraak worden gedaan over de bedrijfszekerheid. Of als bepaalde schermen geen prettige “look&feel” hebben, kan iets gezegd worden over de usability.

3. Uitvoeren statische tests

In de teststrategie is vastgelegd of er statische tests uitgevoerd dienen te worden. Bij statisch testen worden eindproducten beoordeeld zonder dat er sprake is van het runnen van software. Deze test bestaat meestal uit het inspecteren van documentatie zoals beveiligingsprocedures, opleidingen, handleidingen, enzovoort en wordt vaak met checklists ondersteund. Op basis hiervan wordt getracht inzicht te krijgen in het betreffende kwaliteitsaspect. Ook hiervoor geldt dat eventuele bevindingen worden geregistreerd en verwerkt door middel van de bevindingenprocedure (zie hoofdstuk 12 “Bevindingenbeheer”).

Producten

Testresultaten.

Technieken

Exploratory Testing.
Error Guessing.

Tools

Testwarebeheertool.
Bevindingenbeheertool.
‘Model based testing’-tool.

Geautomatiseerde testuitvoering tool.
Performance-, load-, en stresstesttool.
Monitor.
Code coveragetool.
Comparator.
Databasemanipulatietool.
Simulator.
Stubs en drivers.

6.7.4 Controleren en beoordelen testresultaten

Doel

Het vaststellen van de overeenkomsten en het analyseren van de verschillen tussen de verkregen testresultaten en de voorspelde resultaten in de testscripts of checklists.

Werkwijze

De werkwijze omvat de volgende deelactiviteiten:

1. Vergelijken testresultaten;
2. Analyseren verschillen;
3. Bepalen hertests.

1. Vergelijken testresultaten

De testresultaten worden vergeleken met de voorspelde resultaten in de testscripts en checklists. Wanneer er getest is op basis van een explorerende techniek vergelijkt de tester de uitkomst met de gedocumenteerde testbasis als het functioneel ontwerp of een requirementsdocument. Indien er geen gedocumenteerde testbasis is, moet de tester op zoek gaan naar andere manieren om de uitkomst mee te vergelijken. Deze informatie kan bijvoorbeeld gehaald worden uit normen en standaards, memo's, gebruikershandleidingen, interviews, advertenties of concurrerende producten (zie ook de tip "ontbreken van testbasis" in paragraaf 6.5.1. "Verzamelen testbasis).

Uitgediept

Het gevaar van testen zonder gedocumenteerde testbasis

Wanneer geen gedocumenteerde testbasis beschikbaar is voor de tester bestaat het reële gevaar dat de tester gaat vertrouwen op andere informatiebronnen dan de testbasis, zoals zijn of haar intuïtie. Een ongewenst eindresultaat kan dan zijn dat systeem en documentatie niet synchroon lopen. Wanneer het systeem correct is en de documentatie niet, kan dit een onderhouds- of beheerprobleem geven. Andersom is mogelijk (diepe) functionaliteit in de documentatie beschreven die incorrect in het systeem is geïmplementeerd en er bij het testen op basis van andere bronnen dan de systeemdocumentatie niet uit is gekomen. Een ander ongewenst eindresultaat kan zijn dat de testers bij gebrek aan duidelijkheid over de scope een eindeloze stroom wijzigingsverzoeken genereren onder het mom van bevindingen.

Wanneer er geen afwijkingen zijn dan wordt dit gelogd. Indien er wel afwijkingen worden geconstateerd, dan worden deze nader geanalyseerd. Het vergelijken van de testresultaten vindt vaak tegelijk plaats met de uitvoering van de test. Bijvoorbeeld door het afvinken van stappen in het testscript kan worden aangegeven of een testresultaat overeenkomt met het verwachte resultaat. In bepaalde gevallen is het niet mogelijk om dit al tijdens de test te doen (bijvoorbeeld bij batchsystemen waar de output van meerdere testgevallen wordt gepresenteerd).

2. Analyseren verschillen

De geconstateerde verschillen worden tijdens deze deelactiviteit nader geanalyseerd. De tester moet hierbij de volgende stappen doorlopen:

- Verzamel bewijsmateriaal;
- Reproduceer de bevinding;
- Controleer op eigen fouten;
- Bepaal vermoedelijke externe oorzaak;
- Isoleer de oorzaak (optioneel);
- Generaliseer de bevinding;
- Vergelijk met andere bevindingen;
- Schrijf bevindingrapport;
- Laat reviewen.

Deze stappen worden toegelicht in paragraaf 12.2 “Een bevinding doen”. De stappen staan in globale volgorde van uitvoering, maar het is zeer goed mogelijk bepaalde stappen in een andere volgorde of parallel uit te voeren. Als de tester bijvoorbeeld gelijk ziet dat de bevinding al eerder in dezelfde test gedaan is, hoeven alle tussenliggende stappen niet meer doorlopen te worden.

In de testscripts worden de nummers van de bevindingen geregistreerd bij díe testgevallen waar de bevinding is geconstateerd. Op deze wijze is bij een eventuele hertest snel duidelijk welke testacties minimaal opnieuw moeten worden uitgevoerd. Zowel voor het vergelijken van de testresultaten als het analyseren van de verschillen zijn diverse testtools beschikbaar (zie paragraaf 8.5 “Testtools”).

3. Bepalen hertests

Redenen voor het uitvoeren van hertests kunnen geconstateerde bevindingen zijn. Wanneer de oorzaak voor de bevinding een fout in de testuitvoering betreft dan wordt deze betreffende test opnieuw uitgevoerd. Bevindingen die hun oorsprong vinden in een fout testscript of checklist worden opgelost. Hierna wordt het gewijzigde deel van het testscript weer uitgevoerd of de gehele checklist weer doorlopen. Fouten in de testomgeving moeten ook worden opgelost waarna de betreffende testscripts in hun geheel weer worden uitgevoerd.

Fouten in het testobject of de testbasis zullen meestal voor een nieuwe versie van het testobject zorgen. Bij een fout in de testbasis zal het ook zo zijn dat de bijhorende testscripts aangepast moeten worden. Dit levert vaak veel werk op. Wanneer hertests plaatsvinden, is het van belang om vast te stellen op welke wijze deze uitgevoerd moeten worden. Het geheel of gedeeltelijk opnieuw uitvoeren van alle testscripts wordt bepaald door de testmanager in de fase Beheer en is ondermeer afhankelijk van:

- de exit-criteria opgesteld in het testplan;
- de ernst van de bevindingen;
- het aantal bevindingen;
- de mate waarin de eerdere uitvoering van het testscript is verstoord door de bevindingen;
- de beschikbare tijd;
- de risico’s.

Uitgediept

Wanneer opgeloste bevindingen testen?

Bevindingen die zijn opgelost moeten weer getest worden. Deze tests kunnen op twee (uiterst) verschillende momenten worden uitgevoerd.

1. Testen zodra een bevinding is opgelost. Het voordeel is dat de programmeur, die de bevinding heeft opgelost, deze nog vers in zijn geheugen heeft. Hierdoor kan de programmeur snel acteren wanneer de bevinding niet opgelost blijkt te zijn. Nadeel is dat de code vaak gewijzigd, opgeleverd en getest wordt. Hierbij kunnen fouten gemakkelijk gemaakt worden en dat werkt voor de tester minder efficiënt.
2. Het verzamelen van opgeloste bevindingen en deze testen. Het voordeel hiervan is dat bevindingen gegroepeerd (bijvoorbeeld per module of per scherm) opgelost en getest kunnen worden wat een efficiënte werkwijze is. Ook

is de code stabieler waardoor de kans op het terugkomen van een bevinding minimaal is. Nadeel is wel dat deze manier van werken een langere doorlooptijd heeft

De keuze voor optie 1 of 2 hangt af van het project en de manier waarop er gewerkt wordt. Wanneer het mogelijk is om dagelijks een release van een applicatie op te leveren[1] en er een groot aantal te hertesten bevindingen zijn, dan kan het een strategie zijn om een mix van bovenstaande te kiezen. Iedere dag wordt dan bepaald welke opgeloste bevindingen in de release meegaan en deze kunnen dan de volgende dag worden getest door het testteam. Het is dan wel belangrijk om hiervoor een aparte testomgeving te reserveren en de releases alleen te gebruiken voor het testen van de bevindingen. Daarnaast zal na afloop een test moeten plaatsvinden op het gehele testobject om vast te stellen dat er niks anders is gewijzigd (regressie).

Producten

Bevindingen.
Logging van de testresultaten.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool.
Bevindingenbeheertool.
Testdatatool.
Geautomatiseerde testuitvoering tool.
Performance-, load-, en stresstesttool.
Monitor.
Code coveragetool.
Comparator.
Databasemanipulatietool.

6.8 Fase Afronding

Doel

- het leren van de ervaringen die zijn opgedaan in deze test;
- het conserveren van testware om bij een volgende test deze te kunnen hergebruiken.

Context

Bij de gestructureerde testaanpak van TMap kan veel winst worden behaald in de herhaalbaarheid van het proces. Hierdoor kunnen producten, mits ze voldoen aan bepaalde eisen, weer hergebruikt worden in een volgende test. Dit kan er dan voor zorgen dat bepaalde activiteiten sneller kunnen verlopen. Producten kunnen tastbare zaken als testgevallen of testomgevingen zijn, maar ook niet-tastbare zaken als ervaringen worden hiertoe gerekend.

Randvoorwaarden

Aan de volgende voorwaarde dient te zijn voldaan alvorens kan worden gestart met de fase Afronding:

- de testuitvoering is in een afrondend stadium.

Werkwijze

Het testproces wordt geëvalueerd. Doelstelling hiervan is om te leren van de opgedane ervaringen en deze leerpunten toe te passen in een eventuele nieuwe test. Ook dient dit als input voor het eindrapport wat door de testmanager in de fase Beheer wordt opgesteld. Tevens wordt een selectie gemaakt uit de vaak grote hoeveelheid testware, zoals de testgevallen en de beschrijving van de testinfrastructuur. Het oogmerk is hierbij dat bij wijzigingen en de bijbehorende onderhoudstests de testware slechts aanpassing behoeft en er dus niet een compleet nieuwe test moet worden ontworpen. Tijdens het testproces is getracht de testgevallen in overeenstemming te houden met de testbasis en het ontwikkelde systeem. Zo nodig moeten de geselecteerde testgevallen geactualiseerd worden.

Rollen / verantwoordelijkheden

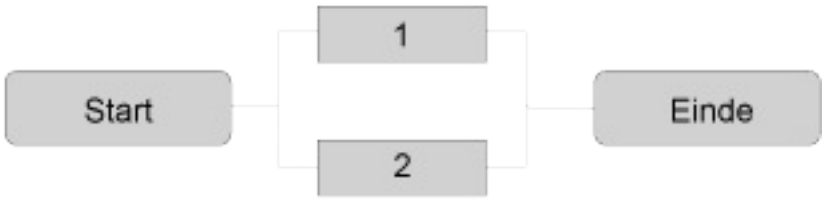
Alle activiteiten kunnen door alle testteamleden uitgevoerd.

Activiteiten

De fase Afronding bestaat uit de volgende activiteiten:

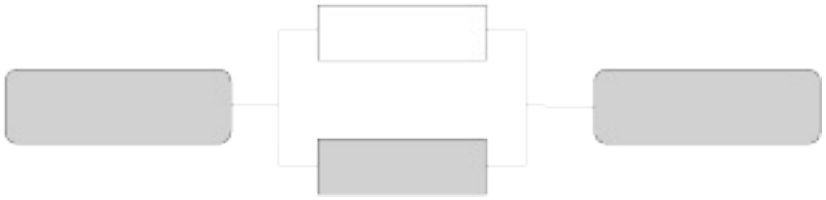
1. Evalueren testproces;
2. Conserveren testware.

Onderstaand schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten:



Figuur 6.20: Afronding.

6.8.1 Evalueren testproces



Doel

Het leren van de ervaringen die zijn opgedaan gedurende de afgelopen test en deze leerpunten zo vast te leggen, dat ze in een volgende test gebruikt gaan worden.

Werkwijze

Het continu leren, en dan vervolgens ook het gebruiken van de nieuwe kennis, is een belangrijk onderwerp in TMap. Een manier om dit te doen is het organiseren van evaluatiesessies. Deze sessies zijn veelal gericht op het genereren van lessen en leerervaringen voor de toekomst. Het onderwerp van evaluatie kan daarbij variëren, al naar gelang de vraagstelling. Het kan gaan om het evalueren van het testproces, de resultaten van de test, de betrokkenheid van de verschillende partijen daarin, het gebruik van de testinfrastructuur, enzovoorts. Van belang hierbij is dat met name duidelijk wordt hoe de betrokkenen het onderwerp ervaren. Een evaluatie moet bij afronding van de test plaatsvinden, maar het is aan te raden dit ook regelmatig tijdens de test zelf te doen. Op deze manier kan continu geleerd worden en het geleerde worden toegepast. Een hulpmiddel bij het stellen van de juiste vragen tijdens een evaluatie is de checklist “evaluatie testproces” (www.tmap.net).

In de fase Beheer wordt door de testmanager een eindrapport opgesteld. In dit eindrapport staat beschreven hoe het testproces is verlopen. Daarnaast worden er ervaringscijfers verzameld ten behoeve van toekomstige testprocessen. Het resultaat van de evaluatie dient hierbij als input.

Tip

Evaluaties als hefboom voor verandering

Het uitvoeren van evaluaties kan een doel hebben dat verder gaat dan het alleen hergebruiken van de opgedane kennis. Het kan ook als doel hebben om de kennis om te zetten tot een hefboom voor veranderingen. Een voorwaarde hiervoor is dat de testmanager in een rol verkeert waarin hij veranderingen kan voorstellen. Deze voorstellen kunnen dan opgenomen worden in het eindrapport. Wanneer dit proces al tijdens de uitvoering van het project plaatsvindt, is het grote voordeel dat de voorgestelde veranderingen meteen doorgevoerd kunnen worden. Voorwaarde voor succes is dat het delen van kennis op alle niveaus moet worden gestimuleerd; dit kan bijvoorbeeld door het organiseren van (informele) bijkomsten. Tijdens deze bijeenkomsten moet er een losse sfeer zijn, waarbij er geen verschillen mogen zijn in rollen en rangen.

Betrek alle partijen bij de meeting, praat over de problemen en probeer meteen een oplossing te vinden.

Voorbeeld

Testers als vraagbaak

Tijdens een test kwamen steeds meer Engelstalige ontwikkelaars langs bij de Nederlandstalige testers om vragen te stellen over bepaalde functionele specificaties. Uit verschillende informele bijeenkomsten op vrijdagmiddag bleek dat zij moeite hadden met de combinatie van het steenkolenengels van de Nederlandse ontwerpers en de lange lappen tekst. De testers hadden vanuit hun expertise diepgaande kennis opgebouwd van het systeem. In overleg met de verschillende partijen is toen besloten dat de testers als vraagbaak mochten dienen voor de ontwikkelaars. Daarnaast is een selectie gemaakt van bestaande testscripts die uitgevoerd moesten worden door de ontwikkelaars voordat ze hun stuk programmatuur zouden opleveren. Dit kwam de algehele kwaliteit van het testproces en snelheid ten goede; bevindingen werden nu al tijdens ontwikkeling gevonden en er ontstond een grotere betrokkenheid van de ontwikkelaars bij het testen en andersom.

Producten

Evaluatie testproces.

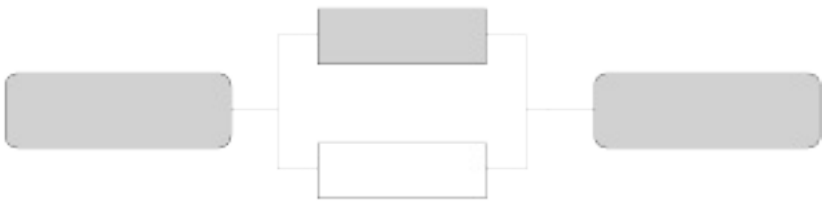
Technieken

Checklist evaluatie testproces (www.tmap.net).

Tools

Testwarebeheertool.
Bevindingenbeheertool.

6.8.2 Conserveren testware



Doel

Het selecteren en actualiseren van de vervaardigde testware, zodanig dat bij toekomstige tests optimaal van deze testware gebruik kan worden gemaakt.

Werkwijze

De werkwijze omvat de volgende activiteiten:

1. Selecteren testware;
2. Verzamelen, bijwerken en toegankelijk maken testware;
3. Overdragen testware.

Deze activiteit heeft een nauwe relatie met de activiteit “Conserveren infrastructuur” in de fase “Inrichting en beheer infrastructuur” (zie paragraaf 6.4.6).

Tip

Eerder beginnen met de activiteit “Conserveren Testware”

Ook al is de activiteit Conserveren Testware de laatste activiteit in het TMap faseringsmodel, aangeraden wordt hier al eerder mee te beginnen. Door in de fase Specificatie al rekening te houden met een mogelijke conserveringsslag kunnen bepaalde standaarden worden ontwikkeld of met bepaalde tools gewerkt worden, waardoor de uiteindelijke conservering sneller zal verlopen. Door bijvoorbeeld meteen te werken met consistente versienummering en een centrale opslag voor alle producten hoeft tijdens deze activiteit niet meer gezocht te worden naar de laatste versie van alle producten. Het gebruik van een testwarebeheertool kan hier bij helpen. De manier waarop dit gebruik van standaarden is ingericht, wordt gedefinieerd tijdens de activiteit “Inrichten beheer” in de fase Planning (zie paragraaf 6.2.12).

1. Selecteren testware

In overleg met de toekomstige beheerder van het systeem wordt geïnventariseerd welke testware aan hem beschikbaar wordt gesteld. Het oogmerk hierbij is, dat bij wijzigingen en de bijbehorende onderhoudstests, de testware herbruikbaar is en er dus geen compleet nieuwe test ontworpen hoeft te worden. De uiteindelijke keuze voor welke testware beschikbaar wordt gesteld, wordt gemaakt op basis van een kosten-baten analyse. Onderwerpen zijn dan “wat kost het om de testware te beheren (opslag en actueel houden)” en “wat kost het om het nieuw te maken”.

De op te leveren testproducten worden vastgelegd in een zogenaamde paklijst. Dit is een overzicht van de te conserveren testproducten. Het is van belang aan te geven op welke wijze deze testproducten tot stand zijn gekomen, om toekomstig onderhoud op de juiste wijze door te kunnen voeren. Hierbij moet met name worden gedacht aan de gehanteerde testontwerptechnieken, tools, enzovoort.

2. Verzamelen, bijwerken en toegankelijk maken testware

De over te dragen testware moet daar waar nodig worden gecompleteerd en aangepast. Met name gedurende de laatste fasen van de uitvoering komt het voor, dat het onderhoud op de testware wordt uitgesteld. Voordat overgedragen kan worden aan de toekomstige gebruikers moeten de eventuele wijzigingen alsnog zijn verwerkt. Ook moet de testware toegankelijk gemaakt worden. Dit betekent dat het op een zodanige wijze opgeslagen dient te worden dat het voor de toekomstige gebruikers handig is. Dat kan bijvoorbeeld betekenen dat de directorystructuur anders opgezet moet worden of er een bepaalde tool gebruikt moet worden.

Uitgediept

Regressietestset aanpassen

Het actueel houden van een regressietestset wil nog wel eens vergeten worden. Het is daarom aan te bevelen om deze activiteit standaard in de activiteit “Conserveren testware” op te nemen. Tijdens de testuitvoering kan het zijn

voorgekomen dat het systeem toch anders reageerde dan in het testscript werd aangenomen. Als deze aanname fout was, moet het testscript conform de nieuwe situatie worden aangepast. Verder moet bepaald worden of er, en zo ja welke, nieuwe testscripts aan de bestaande regressietestset moeten worden toegevoegd.

3. Overdragen testware

Tenslotte vindt de daadwerkelijke overdracht van de testware plaats. Conform de paklijst worden alle geselecteerde delen in digitale vorm en soms ook fysiek (op papier) overgedragen aan de beheerafdeling.

Producten

Paklijst testware.
Herbruikbare testware.

Technieken

Niet van toepassing.

Tools

Testwarebeheertool.
Testdatatool.

7 Ontwikkeltesten

Gastschrijver: Rob Kuijt

7.1 Inleiding

De gebruikers van informatiesystemen eisen, om optimaal in de markt te kunnen acteren, een steeds hogere snelheid van opleveren én meer flexibiliteit van de systemen. De ontwikkelmethoden zijn er steeds meer op gericht om zelfs tijdens een project de veranderende wensen direct te volgen. De architecturen en ontwikkelomgevingen die dit alles mogelijk maken, worden steeds complexer en groter. Dit verandert de eisen aan de ontwikkeling.

De steeds hardere eis aan de ontwikkelaar: op tijd en in één keer de juiste kwaliteit opleveren!

Naast de kennistoename op het gebied van de ontwikkeltalen, -methoden, -omgevingen en -architecturen, vereist dit ook meer kennis over het leveren van kwaliteit. Lastige vragen in dit verband zijn: wat is het voor de klant noodzakelijke kwaliteitsniveau en hoe kan dit kan worden gerealiseerd én aangetoond door middel van testen? Eigen interpretatie van de gewenste kwaliteit en uit de losse hand testen geeft geen garantie voor uiteindelijk succes. Voorspelbare en aangetoonde kwaliteit van de opgeleverde software geeft het project of de afdeling de kans om de aansluitende testsoorten als de systeem- en acceptatietests efficiënter in te richten. Vooral een vermindering van het aantal heropleveringen en hertests in die testsoorten levert een enorme tijdwinst. Om deze hogere softwarekwaliteit te realiseren, worden steeds hogere eisen aan de ontwikkeltests gesteld en groeit het ontwikkeltesten naar een volwassen onderdeel van het totale testproces.

Het einde is daarmee in zicht dat er nauwelijks eisen aan de ontwikkeltests gesteld worden en dat er wordt gerekend op de (steeds volwassener geworden) systeem- en acceptatietests om het gebrek aan kwaliteit voor in-productiename te herstellen. De hieruit voortvloeiende langdurige en kostbare herstel- en hertest-cycli zijn voor de meeste organisaties onacceptabel geworden.

Dit hoofdstuk licht in paragraaf 7.2 de ontwikkeltests toe, maakt een vergelijking met de overige testsoorten en beschrijft specifieke testtools voor het ontwikkeltesten. Ook worden diverse kwaliteitsmaatregelen besproken die gebruikt kunnen worden in of van invloed zijn op de ontwikkeltests. Een belangrijke maatregel hierbij is het concept van afgesproken kwaliteit. Vervolgens beschrijft paragraaf 7.3 de activiteiten van het ontwikkeltesten volgens het TMap-faseringsmodel.

Hoewel de ontwikkelaars/ontwikkeltesters een logische doelgroep van dit hoofdstuk zijn, moeten zij niet verwachten dat ze na het lezen van dit ene hoofdstuk weten hóe ze de ontwikkeltest moeten inrichten en uitvoeren. De doelgroepen van dit hoofdstuk zijn:

- de ontwikkelaar/ontwikkeltester om ideeën op te doen voor een betere ontwikkeltest;
- de testadviseur die gevraagd wordt om (de inrichting van) het ontwikkeltesten te ondersteunen;
- de testmanager voor het overkoepelende testproces die de ontwikkeltests met de andere testsoorten moet afstemmen;
- de lijnmanager van de ontwikkelaars, die geïnteresseerd is in het beter beheersen van de kwaliteit van de geproduceerde software en wil weten hoe dit bereikt kan worden.

7.2 Ontwikkeltesten toegelicht

Deze paragraaf bestaat uit een aantal subparagrafen. Dit zijn achtereenvolgens:

- Wat is ontwikkeltesten
- Kenmerken
  Met focus op de verschillen met systeem- en acceptatietesten;
- Voor- en nadelen van beter ontwikkeltesten
- Context van ontwikkeltesten
  De invloed van de ontwikkelmethode en technische implementatie
- Unittest
- Unitintegratietest
- Kwaliteitsmaatregelen
  Diverse maatregelen, waaronder het concept van afgesproken kwaliteit van ontwikkeltesten

- Testtools voor ontwikkeltests.

7.2.1 Wat is ontwikkeltesten?

Onder ontwikkeltesten wordt verstaan het testen met gebruikmaking van kennis van de technische implementatie van het systeem. Dit begint met het testen van de eerste/kleinste onderdelen van het systeem: *routines, units, programma's, modules, componenten, objecten, enzovoort…* Binnen TMap wordt in dit kader uitsluitend de term unit en dus *unittest* gehanteerd.

Nadat is vastgesteld dat de meest elementaire delen van het systeem van voldoende kwaliteit zijn, worden tijdens de unitintegratietests grotere delen van het systeem integraal getest. Hierbij ligt de nadruk op de gegevensdoorvoer en de interface werking tussen de units tot op deelsysteemniveau.

Definitie

Unittest (UT)

De *unittest* is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een unit aan de in de technische specificaties gestelde eisen voldoet.

Unitintegratietest (UIT)

De *unitintegratietest* is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een logische groep units aan de in de technische specificaties gestelde eisen voldoet.

7.2.2 Kenmerken

Een valkuil bij het inrichten van het ontwikkeltesten is het inrichten van een testproces vanuit de optiek van een systeem- of acceptatietest. Wanneer ontwikkeltests worden vergeleken met de systeemtest en de acceptatietest, is een aantal belangrijke verschillen te constateren:

- In tegenstelling tot de systeemtest en acceptatietest zijn de ontwikkeltests niet te organiseren als een zelfstandig proces met een min of meer onafhankelijk team. De ontwikkeltests maken een integraal onderdeel uit van het ontwikkelen van software. De fasering van de testactiviteiten is dan ook geïntegreerd met de activiteiten van de ontwikkelaars.
- Omdat ontwikkeltesten gebruik maakt van kennis van de technische implementatie van het systeem, worden andersoortige fouten gevonden dan bij systeem- en acceptatietests. Van ontwikkeltests kan (en mag) bijvoorbeeld verwacht worden dat elk statement in de code een keer is doorlopen. Een dergelijke dekkingsgraad is voor systeem- en acceptatietests in de praktijk erg moeilijk te bereiken, deze testsoorten richten zich op andere aspecten. Het is dus niet goed mogelijk om ontwikkeltests te vervangen door systeem- en acceptatietests.
- Met name bij de unittests is vaak de ontdekker van de fouten (=tester) dezelfde persoon als de oplosser van de fouten (=ontwikkelaar). Dit betekent dat de communicatie over de fouten minimaal kan zijn.
- De insteek van het ontwikkeltesten is dat alle geconstateerde fouten zijn opgelost vóórdat de software wordt overgedragen. De rapportage van het ontwikkeltesten kan daarom beperkter zijn dan bij systeem- en acceptatietests.
- Het is het eerste testtraject, wat betekent dat alle fouten nog in het product zitten. Dit vereist goedkope en snelle foutaanpassing. Om dit te realiseren is een flexibele testomgeving met weinig procedurele barrières van groot belang.
- Bij ontwikkeltesten zijn vaak ontwikkelaars aan het testen. De basishouding van een ontwikkelaar is aan te tonen dat het gerealiseerde product werkt, terwijl een tester het verschil tussen de vereiste en actuele kwaliteit van het product wil aantonen (en hiervoor actief op zoek gaat naar fouten). Dit verschil in "mindset" betekent dat omvangrijk en/of diepgaand ontwikkeltesten haaks staat op de basishouding van de ontwikkelaar en daarmee veel weerstand oproept en/of resulteert in onzorgvuldig uitgevoerde tests.

Uitgediept

Onderstaande tabel [Pettichord, 2000] geeft een opsomming van een aantal markante kenmerken van testers en ontwikkelaars:

Testers	Ontwikkelaars

Komen snel op gang	Diepgaand begrip
Domein kennis	Kennis van de interne productstructuur
Een lichte mate van onwetendheid is belangrijk	Deskundigheid is belangrijk
Modelleren gebruikersgedrag	Modelleren systeemontwerp
Gericht op wat mis kan gaan	Gericht op hoe het kan werken
Gericht op ernst van het probleem	Gericht op interesse in het probleem
Empirisch	Theoretisch
Wat is waargenomen	Hoe het is ontworpen
Sceptici	Vertrouwen in eigen kunnen
Tolereren verveling	Automatiseren verveling
Geen problemen met conflicten	Vermijden conflictsituaties
Rapporteren problemen	Begrijpen problemen

Tabel 7.1: Kenmerken van testers en ontwikkelaars

Uitgediept

Onderstaande tabel [Pettichord, 2000] geeft een opsomming van een aantal markante kenmerken van testers en ontwikkelaars:

Testers	**Ontwikkelaars**
Komen snel op gang	Diepgaand begrip
Domein kennis	Kennis van de interne productstructuur
Een lichte mate van onwetendheid is belangrijk	Deskundigheid is belangrijk
Modelleren gebruikersgedrag	Modelleren systeemontwerp
Gericht op wat mis kan gaan	Gericht op hoe het kan werken
Gericht op ernst van het probleem	Gericht op interesse in het probleem
Empirisch	Theoretisch
Wat is waargenomen	Hoe het is ontworpen
Sceptici	Vertrouwen in eigen kunnen
Tolereren verveling	Automatiseren verveling
Geen problemen met conflicten	Vermijden conflictsituaties
Rapporteren problemen	Begrijpen problemen

Tabel 7.1: Kenmerken van testers en ontwikkelaars

7.2.3 Voor- en nadelen van beter ontwikkeltesten

In de praktijk verloopt het ontwikkeltesten vaak ongestructureerd: tests worden niet gepland of voorbereid, er wordt geen gebruik gemaakt van testontwerptechnieken en er is geen inzicht in wat nu wel of niet getest is en hoe zwaar. Daarmee ontbreekt tevens inzicht in de kwaliteit van het (geteste) product. Tijdens de systeem- en acceptatietests ontstaan dan vaak langdurige en inefficiënte test/herstel/hertestcycli om de kwaliteit alsnog op een acceptabel niveau te krijgen. Het ligt dus voor de hand om het ontwikkeltesten beter in te richten. Onderstaand wordt een aantal argumenten gegeven waarom dit in de praktijk niet gebeurt (argumenten tegen) én waarom het zo belangrijk is dat het wel gebeurt (argumenten voor).

Argumenten tegen

De belangrijkste argumenten waarom meer structuur en diepgang in het ontwikkeltesten *niet* vanzelfsprekend zijn:

- *Tijdsdruk / te duur ten opzichte van baten*
  Vaak staan de ontwikkelaars onder grote tijdsdruk. De prioriteit van het ontwikkelteam ligt op de criteria waar het op wordt beoordeeld. Er wordt meestal beoordeeld op een hard criterium als tijd en geleverde functionaliteit. Beoordeling op een veel zachter criterium als kwaliteit is moeilijker en vindt daarom in de praktijk weinig plaats. Een ontwikkelaar die commitment heeft gegeven op een einddatum, zal óf open en eerlijk communiceren dat het tegenzit, óf zijn eigen testen minder tijd geven als het coderen tegenzit. In het licht van persoonlijk functioneren (en beoordeling) is het laatste niet ondenkbaar. De baten van goed testen zijn in zo'n geval voor het ontwikkelteam vrij klein, ook al zijn de baten van goed testen voor het gehele project vele malen groter.
- *Voldoende vertrouwen in de kwaliteit*
  Een ontwikkelaar is normaal gesproken trots op zijn product en vindt het kwalitatief goed. Het is daarom onlogisch om

als ontwikkelaar veel moeite te doen om fouten in het eigen product te vinden.
- *Er volgt nog een goede test*
  In het volgende traject, bijvoorbeeld de systeemtest, wordt een veel intensievere test uitgevoerd dan het ontwikkeltesten ooit kan, zal of wil doen. Waarom zou de ontwikkeltester dan nog veel aandacht besteden aan meer en beter testen, als het later toch uitgebreider gebeurt?

Argumenten vóór

Het belangrijkste argument voor meer structuur en diepgang in het ontwikkeltesten is, dat hiermee de ontwikkelaar voor zichzelf kan vaststellen dat de software van voldoende kwaliteit is om opgeleverd te worden aan het volgende traject, waarschijnlijk de systeemtest. Over de betekenis van “voldoende kwaliteit” is natuurlijk discussie mogelijk. Onderstaand wordt aangegeven dat “voldoende kwaliteit” vele voordelen heeft *voor het ontwikkelteam*:

- Er zal na oplevering minder herstelwerk nodig zijn, omdat de producten die aan volgende trajecten worden opgeleverd, van hogere kwaliteit zijn.
- De planning wordt beter, omdat de vaak onzekere hoeveelheid herstelwerk afneemt.
- De doorlooptijd van het totale ontwikkeltraject wordt, om dezelfde reden, korter.
- Zo vroeg mogelijk herstellen is veel goedkoper dan in een later stadium herstellen, omdat alle kennis over de ontwikkelde producten nog vers in het geheugen ligt en omdat in latere stadia vaak al mensen het ontwikkelteam hebben verlaten.
- De analyse van zelf gevonden fouten is veel sneller en makkelijker dan de analyse van fouten die door anderen gevonden zijn. Hoe meer afstand (zowel organisatorisch als fysiek) de vinder van de fout heeft, des te moeilijker en tijdrovender is vaak de analyse. Dit wordt nog versterkt omdat in latere trajecten het systeem als geheel wordt getest, zodat de gevonden fout vaak in vele afzonderlijke componenten kan zitten.
- De ontwikkelaars krijgen sneller terugkoppeling van hun gemaakte fouten, zodat ze eerder en beter in staat zijn dergelijke fouten in andere units te voorkomen.
- Bepaalde fouten, met name op de grensvlakken van systeemfunctionaliteit en onderliggende besturingssysteem, database en netwerk, zijn het beste met ontwikkeltests te detecteren. Wanneer het ontwikkeltesten een te klein deel van deze fouten vindt, heeft dit gevolgen voor de systeem- en acceptatietests. Deze moeten dan met gebruikmaking van inefficiënte technieken onevenredig (voor detectie van dergelijke fouten) veel inspanning leveren om dezelfde kwaliteit van het testobject te bereiken als wanneer de ontwikkeltests goed waren uitgevoerd.

Voor het totale project en zelfs voor de totale levenscyclus van het systeem gelden deze voordelen *in versterkte mate*, omdat de latere testsoorten ook (en vaak zelfs meer!) van deze voordelen profiteren, bijvoorbeeld omdat veel minder hertests nodig zijn.

Hiermee wegen de voordelen van een meer gestructureerde ontwikkeltestaanpak ruimschoots op tegen de nadelen. Wel is een noodzakelijke voorwaarde voor geslaagde structurering van het ontwikkeltesten dat de diverse betrokkenen, zoals de opdrachtgever, de projectmanager en de ontwikkelaars, zich bewust zijn van het belang van een beter testproces. Zo dient de projectmanager het ontwikkelteam veel meer op geleverde kwaliteit te beoordelen dan alleen op tijd en geld. Ook kan de ontwikkelafdeling eisen stellen aan alle uitgevoerde ontwikkeltests. Elke ontwikkeltest in een individueel project moet dan minimaal aan deze eisen voldoen. Dit wordt toegelicht in paragraaf 7.2.7 “Kwaliteitsmaatregelen”.

7.2.4 Context van ontwikkeltesten

Het ontwikkeltesten heeft een zeer nauwe relatie met het ontwikkelproces en kan eigenlijk niet los daarvan gezien worden. Ook is voor ontwikkeltesten veel meer kennis nodig van de technische implementatie van het systeem of pakket dan voor een systeem- of acceptatietest. Om het ontwikkeltesten goed in te richten, moet daarom sterk rekening worden gehouden met het gehanteerde ontwikkelproces en de technische implementatie.

Tip

Zorg ervoor dat je als adviseur of testmanager bij het inrichten van het ontwikkeltesten voldoende kennis hebt van het gehanteerde ontwikkelproces en de technische implementatie. Dit maakt je tevens, zonder expert te hoeven zijn, een zinvolle gesprekspartner voor de ontwikkelaars.

Invloed van de ontwikkelmethode

Er zijn grofweg 3 stromingen van ontwikkelmethoden te onderscheiden: waterval, iteratief en agile.

- Waterval, met onder andere de kenmerken: het in één keer bouwen van het systeem, gefaseerd met duidelijke overdrachtsmomenten, vaak een langdurig cyclisch proces (o.a. SDM).
- Agile, wel vertaald als lichtvoetig, gekenmerkt door de vier principes: individuen en interactie boven processen en tools, werkende software boven uitgebreide systeemdocumentatie, gebruikersinbreng boven contractonderhandeling, reageren op veranderingen boven het volgen van een plan (o.a. eXtreme Programming en SCRUM).
- Iteratief, met onder andere de kenmerken: het in gedeelten (increments) bouwen van het systeem, gefaseerd met duidelijke overdrachtsmomenten; kort cyclisch proces (iteraties), (o.a. DSDM en RUP). Iteratieve methoden houden het midden tussen waterval en agile.

Om te weten wat de invloed is van de ontwikkelmethode op (de inrichting van) het ontwikkeltesten, dient gekeken te worden in welke mate onderstaande aspecten een rol spelen:

- Voorschriften voor ontwikkeltestactiviteiten;
  Veel methoden gaan niet verder dan aangeven dat er ontwikkeltests uitgevoerd dienen te worden. Zelden worden er gestructureerde richtlijnen gegeven. Extreme Programming (XP), als één van de agile methoden, is een positieve uitzondering op dit vlak. Drie van de voor ontwikkeltesten belangrijkste practices zijn: Pair Programming, Test-Driven Development en Continuous Integration. Verderop in paragraaf 7.2.7 “Kwaliteitsmaatregelen” worden deze practices kort toegelicht.
- Kwaliteit van de testbasis;
  Bij waterval is deze veelal in een formeel beschreven vorm vastgelegd. Bij iteratieve en agile ontwikkelmethoden is de vorm van de testbasis veel minder formeel en vaak ook mondeling overeengekomen (door overleg met gebruikers). Dit betekent dat het bij iteratieve en agile methoden lastiger is te achterhalen wat allemaal getest moet worden. Zo blijven de foutafhandeling en uitzonderingssituaties (samen naar schatting toch snel 80% van de code) vaak onderbelicht in dergelijke vormen van testbasis. Er wordt een groter beroep op de kunde en creativiteit van de ontwikkeltesters gedaan om ook hiervoor tests te bedenken en uit te voeren.
- Lang- of kort-cyclische ontwikkeling.
  Bij kort-cyclische ontwikkeling wordt naar verhouding meer tijd aan testen besteed, met name door de noodzaak om veel frequenter (minimaal elke cyclus) een regressietest op het tot dan ontwikkelde systeem uit te voeren.

Invloed van de technische implementatie

In de loop der jaren is de IT-wereld uitgegroeid tot een bonte lappendeken van technologische oplossingen. Als men dit zeer beknopt wil schetsen, kan gezegd worden dat de eerste systemen monolithisch werden opgezet, wat wil zeggen dat de presentatie, de applicatielogica en de informatieopslag één geheel vormen. Sommige van deze systemen zijn al meer dan 30 jaar in operatie. De monolithische systemen zijn opgevolgd door systemen op basis van client/server-architecturen. Hierna kwamen de 3-lagen systemen met aparte lagen voor presentatie, applicatielogica en database. Parallel hieraan mag de opmars van de grote pakketsoftware zoals SAP bekend verondersteld worden, evenals de opkomst van internet en browser-based applicaties. Tegenwoordig worden veel systemen gedistribueerd opgezet, wat inhoudt dat deze bestaan uit verschillende en vaak fysiek verspreide componenten of services, maar dat het systeem middels een hechte samenwerking naar de buitenwereld toe een samenhangend geheel is.

De systemen zijn ontwikkeld met behulp van een groot arsenaal aan al of niet objectgeoriënteerde programmeertalen, in ontwikkelomgevingen die het testen in meer of mindere mate (geautomatiseerd) ondersteunen.

Uitgediept

Steeds vaker wordt bij systeemontwikkeling gebruik gemaakt van zogenaamde services. Een service in dit verband kan gezien worden als een ‘zelfstandige’, herhaalbare bedrijfstaak, zoals “controleer kredietwaardigheid” en “open nieuwe rekening”. Services zijn opgezet als modulaire en veelal kleine applicaties die gevonden, benaderd en in gebruik kunnen worden genomen door andere applicaties, zonder dat hiervoor kennis nodig is over de onderliggende technische implementatie van de service.

Een Service-Oriented Architectuur (SOA) is een raamwerk voor het integreren van bedrijfsprocessen en ondersteunende IT infrastructuur als veilige, gestandaardiseerde componenten – services – die hergebruikt en

gecombineerd kunnen worden om tegemoet te komen aan veranderende bedrijfsbehoeften [Bieberstein, 2005].

Zoals al in hoofdstuk 3 “TMap in essenties” is aangegeven, is testen een risicogebaseerde activiteit, waarbij risico = faalkans x schade, met faalkans = frequentie van gebruik x foutkans. De relevantie van bovenstaande “samenvatting van 50 jaar systeemontwikkeling in één alinea” is dat de technische implementatie in sterke mate bepaalt wat voor soorten fouten gemaakt kunnen worden en in welke delen de foutkans het grootst is. De teststrategie van het ontwikkeltesten is daarmee sterk afhankelijk van de technische implementatie, méér dan de systeem- en acceptatietest waar meer gekeken wordt naar de specificaties van het systeem en de potentiële schade.

Uitgediept

Het toenemend gebruik van gedistribueerde systemen met grote aantallen componenten en services stelt hoge eisen aan de kwaliteit van de individuele component of service. De complexe samenwerking tussen al deze componenten en services maakt het vinden van de oorzaak van een fout zeer moeilijk en tijdrovend. Het integreren van kwalitatief onvoldoende componenten of services in het systeem en hopen dat de fouten bij systeem- of acceptatietesten wel gevonden worden, heeft tot gevolg dat de gewenste systeemkwaliteit (op tijd en binnen budget) niet geleverd kan worden. De technische aard van veel componenten en services betekent dat de ontwikkeltests een zwaardere verantwoordelijkheid krijgen om vast te stellen dat de afzonderlijke componenten of services van voldoende kwaliteit zijn vóórdat deze geïntegreerd worden.

7.2.5 Unittest

Bij unittesten is het belangrijk om te realiseren wat de plaats van het testen binnen het ontwikkelen is, zoals beschreven in paragraaf 7.2.2. De unittesters zijn meestal de ontwikkelaars die hun eigen unit testen. De projectleider ontwikkeling, een aparte testcoördinator of de in paragraaf 7.2.7 beschreven applicatie-integrator coördineren de tests.

Een aandachtspunt is het specificeren van testgevallen. Ontwikkelaars zien hier niet altijd het nut van in. Door waar mogelijk de keuze te maken voor lichte testontwerptechnieken en met name voor lichte vormen van testdocumentatie, wordt de acceptatiegraad aanzienlijk verhoogd. Vooral bij handmatige unittests is de nodige overtuigingskracht vereist om de ontwikkelaars te laten inzien dat het uitschrijven van testgevallen, in die specifieke gevallen, voordelen biedt op het onvoorbereid uitvoeren van de tests.

Uitgediept

Een goed voorbeeld waaruit het voordeel van testontwerptechnieken en het specificeren van testgevallen blijkt, is het testen van een meervoudige conditie (ALS A=1 en B=2 en C=3 DAN …). Met behulp van de testontwerptechniek Elementaire Vergelijkingen Test (EVT) kan relatief eenvoudig een beperkte set van testgevallen (4 in dit voorbeeld) worden afgeleid die een hoge testdekking geeft. Het zonder techniek bedenken van testgevallen leidt hier al snel tot of een te magere testdekking of juist een veelvoud aan testgevallen (8 in dit voorbeeld).

Steeds meer ontwikkelomgevingen maken het tegenwoordig mogelijk om de (geautomatiseerde) testcode bij de (source)code op te nemen. De unittest bestaat dan uit het starten van de testcode, die vervolgens (een gedeelte van) de sourcecode uitvoert. Dergelijke unittests worden gegroepeerd in een zogenaamd testraamwerk.

Definitie

Een *testraamwerk* (test harness) is een voor een ontwikkelomgeving geconfigureerde verzameling software en testgegegevens om één unit of serie van units dynamisch te testen waarbij het gedrag en de output wordt gecontroleerd.

Het schrijven van unittests in een testraamwerk is een extra inspanning die niet genegeerd mag worden. De ervaring leert dat het ontwikkelen van testcode 10%-30% extra inspanning kost [Vaaraniemi, 2003].

De ontwikkelmethoden hebben de mogelijkheden om testcode direct bij de (source)code op te nemen stevig omarmd. Initiatieven als Test Driven Development (zie paragraaf 7.2.7) maken het testen tot een steeds belangrijker onderdeel van de systeemontwikkeling.

7.2.6 Unitintegratietest

Wanneer een unit is getest en goed bevonden, kan deze geïntegreerd worden met andere units tot een werkend (deel van het) systeem. Vrijwel nooit worden alle units in één keer samengevoegd en getest, het zogenaamde “big bang” scenario. Het nadeel van deze late integratie is dat er in het algemeen veel fouten worden gevonden en dat het traceren van de foutoorzaak veel tijd vraagt.

Een effectievere werkwijze is dat telkens een aantal units met elkaar worden geïntegreerd en dat er na iedere integratiestap wordt getest. Fouten worden zo in een vroegtijdig stadium gevonden wanneer de foutoorzaak nog relatief eenvoudig is te achterhalen. Het unitintegratietesten heeft daarmee vooral een rol in het keer op keer aantonen dat de nieuwe of aangepaste unit(s) in samenhang met eerder geïntegreerde units goed (blijven) werken.

Wat de beste integratievolgorde is en hoeveel integratiestappen noodzakelijk zijn, is afhankelijk van de plaats van de meest risicovolle delen van het systeem. Bij voorkeur begint de integratie met die units waarin de meeste problemen worden verwacht. Hierdoor wordt voorkomen dat aan het eind van de unitintegratietest nog grote problemen naar voren komen.

Uitvoeren van unitintegratietests vereist veel kennis van de inhoud, structuur en vooral de informatie-uitwisseling van onderliggende units. Deze diepgaande kennis maakt dat er aan het integreren van units vaak een aparte rol wordt toegekend: de applicatie-integrator (zie paragraaf 7.2.7).

De ontwikkelingen op het gebied van ontwikkelomgevingen maken het ook mogelijk om geautomatiseerd te compileren, te integreren en te testen. Dit gebeurt met hulp van zogenaamde build & deploy scripts. Build is in deze betekenis het samenvoegen van de verschillende softwarecomponenten tot een zogenaamde software package die geëxporteerd kan worden naar een bepaalde omgeving. Deploy is het uitrollen van de software in de doelomgeving, met andere woorden het omzetten van de software package naar de operationele (geïnstalleerde) vorm. Scripts maken het mogelijk de build en deploy geautomatiseerd uit te voeren. Binnen de build & deploy scripts wordt het testraamwerk aangeroepen. Hierin kunnen, naast de geautomatiseerde unittests, ook tests gebouwd en uitgevoerd worden die de grenzen van de units overschrijden en de integratie testen. Integratietestgevallen vormen vaak een functioneel pad van begin tot eind door de applicatie heen.

Door gebruik te maken van stubs en drivers (zie paragraaf 8.5.3 “Soorten testtools”) zijn in een vroeg stadium tests op te nemen die de applicatie van begin tot het eind doorlopen.

Deze mogelijkheid om geautomatiseerd te integreren en te testen heeft – net als bij geautomatiseerd uitvoeren van unittests – zijn weg gevonden in de ontwikkelmethoden. In plaats van de integratie(test) te zien als een vooral afsluitende activiteit is de Continuous Integration-werkwijze (zie paragraaf 7.2.7) geïntroduceerd, die mogelijke problemen bij het combineren van units zover mogelijk naar voren trekt.

7.2.7 Kwaliteitsmaatregelen

In deze paragraaf staat een aantal maatregelen beschreven die gebruikt kunnen worden in of van invloed zijn op het ontwikkeltesten. De meeste van deze maatregelen hebben gevolgen voor de wijze waarop de unittest en/of de unitintegratietest worden uitgevoerd, behalve de codereview die aanvullend is op de ontwikkeltests en hier geen rechtstreekse invloed op heeft. Het is afhankelijk van de situatie óf en zo ja, welke van deze maatregelen men wil kiezen. Om die reden maken ze geen onderdeel uit van het in paragraaf 7.3 besproken generieke ontwikkeltestactiviteiten, maar worden ze onderstaand kort toegelicht. Het feit dat ze optioneel zijn, wil nadrukkelijk niet zeggen dat de maatregelen maar beperkte voordelen hebben. Integendeel, het goed toepassen van de maatregelen in de juiste context kan zeer grote voordelen opleveren.

Achtereenvolgens worden de volgende maatregelen besproken:

- Test-Driven Development (TDD)

TDD is een ontwikkelmethode die sterk de UT beïnvloedt, omdat het geautomatiseerde tests veronderstelt en ervoor zorgt dat voor alle (source)code testcode aanwezig is.
- Pair Programming
  Een ontwikkelmethode waarbij twee ontwikkelaars aan dezelfde software werken én dus ook in onderlinge samenwerking de unittest specificeren en uitvoeren.
- Codereview
  Deze toets op de code is aanvullend op de ontwikkeltests.
- Continuous Integration
  Een integratieaanpak die geautomatiseerde unit- en unitintegratietests vereist, waardoor de kans op regressiefouten wordt geminimaliseerd.
- Afgesproken kwaliteit van ontwikkeltesten
  Deze aanpak hangt sterk samen met en is onderdeel van de teststrategie. De gemaakte keuzes hebben sterke invloed op de werkwijze van specificeren en uitvoeren van de UT en UIT.
- Applicatie-integrator aanpak
  Een organisatorische oplossing om een hogere kwaliteit van de UT en UIT te bereiken.

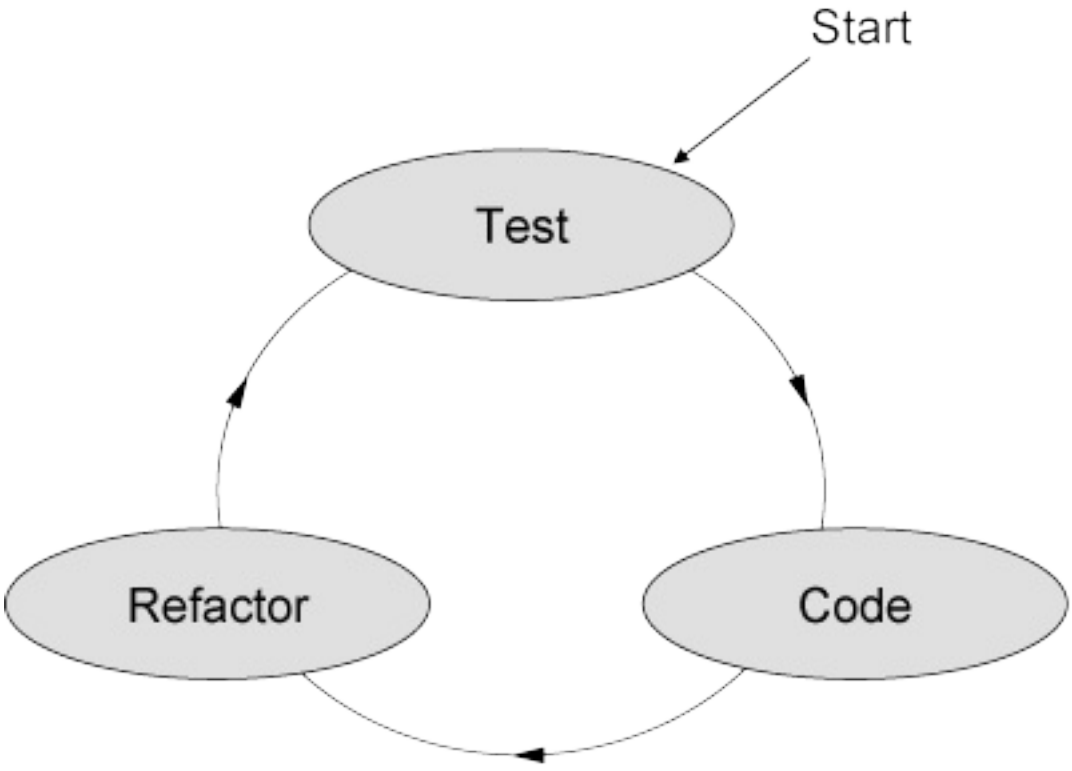
Test-Driven Development (TDD)

Test-Driven Development (testgedreven software ontwikkeling), afgekort TDD, is één van de best practices van Extreme Programming (XP). TDD heeft veel invloed op de manier waarop ontwikkeltesten (ook buiten XP, met name bij iteratieve en agile ontwikkeling) tegenwoordig wordt ingericht. Het is een iteratieve en incrementele wijze van software ontwikkelen waarbij geen code wordt geschreven voordat er geautomatiseerde tests voor die code zijn geschreven. Het doel van TDD is snelle terugkoppeling te bereiken van de kwaliteit van de unit.

Er wordt ontwikkeld in korte cycli volgens bovenstaand schema:

Test maken

- Schrijf een test;
  TDD begint altijd met het schrijven van testcode die een bepaalde eigenschap van de unit controleert.
- Laat de test (fout)lopen;
  Voer de test uit en controleer dat het resultaat van de test negatief is (er is immers nog geen code voor dat stukje functionaliteit).



Figuur 7.1: Test-Driven Development.

Coderen en testen

- Schrijf de code;
  Schrijf de minimaal benodigde code die ervoor zorgt dat de test slaagt. De nieuwe, in dit stadium geschreven code, zal niet perfect zijn. Dat is aanvaardbaar aangezien de volgende stappen het zullen verbeteren. Het is belangrijk dat de geschreven code slechts wordt ontworpen om de test te laten slagen, er mag geen code worden toegevoegd waarvoor geen test is ontworpen.
- Voer de test en alle eerder gemaakte tests uit;
  Als alle testgevallen nu slagen, weet de ontwikkelaar dat de code aan de in test opgenomen requirements voldoet.

Refactoring

- Code opschonen en structureren;
  De code wordt opgeschoond en gestructureerd zonder de semantiek te veranderen. Hierbij voert de ontwikkelaar regelmatig alle testgevallen uit. Zolang deze goed verlopen weet de ontwikkelaar dat zijn aanpassingen geen bestaande functionaliteit beschadigen. Is het resultaat van een testgeval negatief, dan heeft hij een verkeerde wijziging gedaan.

De toepassing van TDD heeft een aantal grote voordelen. Het leidt tot:

- Beter testbare software;
  De code is in de meeste gevallen rechtstreeks te relateren aan een test. Bovendien kunnen de geautomatiseerde tests zo vaak als nodig herhaald worden.
- Een verzameling tests die meegroeit met de software;
  De testware is altijd up-to-date omdat deze 1-op-1 aan de software is verbonden.
- Hoge testdekking;
  Elk stuk code wordt gedekt door een testgeval.
- Beter aanpasbare software;
  Het vaak en geautomatiseerd uitvoeren van de tests geeft zeer snelle terugkoppeling aan de ontwikkelaar of een aanpassing goed of verkeerd is doorgevoerd.
- Hogere kwaliteit van de code;
  De hogere testdekking en het vaak herhalen van de test zorgen dat de software minder fouten bevat bij overdracht.
- Up-to-date documentatie in de vorm van tests.
  De tests maken duidelijk wat het resultaat van de code moet zijn, zodat later onderhoud sterk vergemakkelijkt wordt.

Tip

De theorie van TDD klinkt eenvoudig, maar conform de TDD principes werken vereist grote discipline omdat het gemakkelijk is om ‘uit te glijden’ en weer terug te vallen in de oude gewoonte: functionele code schrijven zonder vooraf een nieuwe test te hebben geschreven. Eén manier om dit te voorkomen is het combineren van TDD en het verderop besproken Pair Programming, de ontwikkelaars houden elkaar dan bij de les.

TDD kent de volgende voorwaarden:

- Een andere manier van denken: veel ontwikkelaars gaan ervan uit dat de extra inspanning voor het schrijven van geautomatiseerde tests niet opweegt tegen de voordelen ervan. Inmiddels heeft de praktijk bewezen dat de extra tijd van testgedreven ontwikkelen ruimschoots wordt goedgemaakt door de besparingen bij het debuggen, wijzigingen van de software en het regressietesten.
- Een tool of testraamwerk voor het maken van geautomatiseerde tests. Voor de meeste programmeertalen zijn testraamwerken beschikbaar (zoals JUnit voor Java en NUnit voor C#).
- Een ontwikkelomgeving die het in korte cycli doorlopen van test-code-refactoring ondersteunt.
- Management commitment om de ontwikkelaars voldoende tijd en gelegenheid te geven om ervaring op te doen met TDD.

Voor meer informatie over TDD wordt verwezen naar [Beck, 2002]

Uitgediept

TDD strategie voor testen van GUI

TDD kent ook zijn beperkingen. Een gebied waar geautomatiseerde unittests moeilijk te implementeren zijn, zijn de Graphical User Interfaces (GUI). Om toch zoveel mogelijk van de voordelen van TDD gebruik te maken kan de volgende strategie worden gevolgd:

- Verdeel de code zoveel mogelijk in componenten die afzonderlijk kunnen worden gebouwd, getest, en geïmplementeerd.
- Houd het grootste deel van de functionaliteit (bedrijfslogica) buiten de context van de GUI. Laat de GUI code een dunne laag boven de door middel van TDD streng geteste code zijn.

Pair Programming

Pair Programming is een andere best practice van Extreme Programming (XP) die ook buiten XP populair is. Bij Pair Programming werken twee ontwikkelaars samen aan hetzelfde algoritme, ontwerp of stuk code, zij aan zij bij één werkplek.

Er is sprake van een duidelijke rolverdeling. De eerste ontwikkelaar is degene die het toetsenbord bedient en de code daadwerkelijk schrijft. De tweede ontwikkelaar controleert (toetst!) en denkt vooruit. Terwijl de code geschreven wordt, denkt hij na over vervolgstappen. Fouten worden snel opgemerkt en verwijderd. Regelmatig wisselen de twee ontwikkelaars van rol.

De significante voordelen van Pair Programming zijn:

- veel fouten worden afgevangen tijdens het typen, in plaats van tijdens de tests of in gebruik;
- het aantal fouten in het eindproduct is statistisch lager;
- de (technische) ontwerpen zijn beter en het aantal coderegels is lager;
- het team lost sneller problemen op;
- de teamleden leren beduidend meer, over het systeem en over softwareontwikkeling;
- het project eindigt met meer teamleden die alle onderdelen van het systeem begrijpen;
- de teamleden leren samenwerken en spreken elkaar vaker, met als gevolg betere informatiestromen en teamdynamiek;
- de teamleden genieten meer van hun werk.

Deze werkwijze is in de laatste decennia verscheidene keren genomineerd als betere manier om software te ontwikkelen. Onderzoek heeft aangetoond dat de kosten door het inzetten van een tweede ontwikkelaar bij Pair Programming niet met 100% stijgen, zoals zou kunnen worden verwacht, maar slechts met ongeveer 15% [Cockburn, 2000]. De investering verdient zich terug in de latere fasen als gevolg van het korter en goedkoper testen, QA en beheer.

Codereview

Een volgende maatregel om de kwaliteit van de ontwikkelde producten te verhogen is een toetsactiviteit: de *codereview*.

Definitie

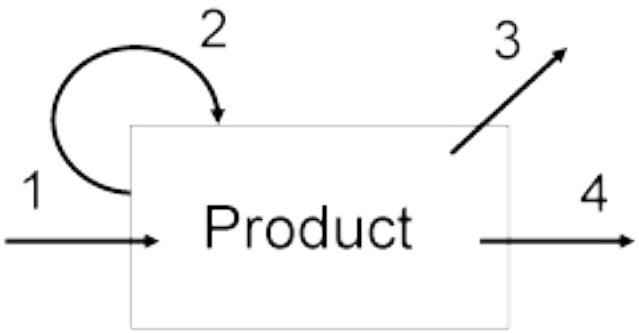
De codereview is een methode om de kwaliteit van geschreven code te verbeteren door het werk te toetsen aan de specificaties en/of richtlijnen en te onderwerpen aan de kritische blik van een aantal gelijken van de ontwikkelaar.

De codereview kan uitgevoerd worden als een statische testactiviteit binnen het ontwikkeltesten. Het doel van de codereview is zekerstellen dat de kwaliteit van de code voldoet aan de gestelde functionele en niet-functionele eisen.

In de codereview kan, afhankelijk van de gestelde eisen, controle plaatsvinden op de volgende punten, zie ook figuur 7.2:

1. Is het product gerealiseerd conform opdracht? Zijn bijvoorbeeld de in het technisch ontwerp vastgelegde eisen juist, volledig en aantoonbaar gerealiseerd?
2. Voldoet het product aan de volgende criteria: intern consistent, conform standaards en normen en de best mogelijke oplossing? ‘Best mogelijke oplossing’ staat voor de ‘beste oplossing’ die binnen gegeven randvoorwaarden als tijd en geld gevonden kan worden.

3. Draagt het product bij aan de project- en architectuurdoelstellingen? Is het product consistent met andere, samenhangende producten (consistentie over de producten heen)?
4. Is het product geschikt voor gebruik in de volgende fase van de ontwikkeling (de integratie)?



Figuur 7.2: De 4 soorten reviewdoelen en -vragen.

Voor de review kunnen verschillende technieken gehanteerd worden, zie ook hoofdstuk 15 “Toetstechnieken”.

Tip

De codereview is ook toepasbaar op ontwikkelomgevingen en -tools waarin vooral geconfigureerd (in plaats van gecodeerd) wordt (bijvoorbeeld bij pakketimplementaties). In dat geval zijn de parameterinstellingen het onderwerp van de review.

Bij het inzetten van de codereview dient rekening gehouden te worden met eventuele overlap met inzet van codeanalysetools, zie paragraaf 7.2.8. Deze tools worden steeds vaker opgenomen in het geautomatiseerde integratieproces, waarbij ze allerlei statische analyses en controles geautomatiseerd uitvoeren. De codereview hoeft hier dus niet meer op te letten. Het is aan te bevelen om de output van de tools in de rapportage van de codereview mee te nemen.

Continuous Integration

Continuous Integration (ononderbroken integratie) heeft vooral invloed op de inrichting van de unitintegratietests. Het is een manier van werken waarbij de ontwikkelaars hun werk regelmatig integreren, minimaal dagelijks en oplopend tot meerdere integraties per dag. De integratie zelf, bestaande uit het samenvoegen van de units en het compileren en linken tot software, is geautomatiseerd. Elke integratie wordt geverifieerd door uitvoering van de geautomatiseerde tests om integratiefouten zo snel mogelijk te ontdekken. De werkwijze minimaliseert de kans op regressiefouten. Continuous Integration is uitstekend te combineren met Test Driven Development, maar vereist een ontwikkelomgeving die het geautomatiseerde integreren en testen ondersteunt.

Afgesproken kwaliteit van ontwikkeltesten

Een belangrijke reden om te ontwikkeltesten is het voldoen aan de vanzelfsprekende verwachting van de ontvangende partij (de opdrachtgever, het project, de systeemtest, …) dat de ontwikkelde software “gewoon” werkt. Als in de opgeleverde software veel fouten optreden, kost dit (veel) geld en tijd om op te lossen. Hier krijgen de ontwikkelaars de schuld van met het verwijt van onprofessioneel handelen.

Maar wat is voor de opdrachtgever(s) eigenlijk vanzelfsprekende kwaliteit? Bij de planvorming voor het inrichten van het ontwikkeltesten wordt er goed aan gedaan om deze, zelden uitgesproken, verwachtingen expliciet te maken. Deze zijn grofweg in te delen in vanzelfsprekende verwachtingen ten aanzien van vakmanschap en vanzelfsprekende verwachtingen met betrekking tot productkwaliteit.

Uitgediept

Om de inventarisatie te vereenvoudigen wordt hieronder een opsomming gegeven van mogelijke verwachtingen.

Vanzelfsprekende kwaliteit van het product:

- Goed is goed genoeg;
  Het opgeleverde product hoeft niet perfect te zijn, maar moet goed genoeg zijn om aan de volgende fase (van testen) te worden overgedragen. Wat vaak gebeurt, is dat ontwikkelaars de eerste units (te) grondig testen. Bij latere units komen ze dan in tijdproblemen en testen ze onvoldoende grondig.
- Eenmaal goed blijft goed;
  Wijzigingen op het product mogen niet tot lagere kwaliteit van het totale product leiden, dus regressiefouten worden niet getolereerd.
- Verwerking van de meest normale gevallen werkt foutloos;
- Basis gebruikersvriendelijkheid (bijv. standaard validaties, technische consistentie, uniformiteit).

Vanzelfsprekend vakmanschap:

- Basiskennis projectmatig werken;
- Ken de opgeleverde kwaliteit;
- Haalplicht om bevestiging te krijgen van de aannames (interpretaties van de specificaties);
- Brengplicht “known errors” en/of “melding uitloop”;
- Bewust van eigen onervarenheid en/of onbekwaamheden;
- Optimale inzet van de beschikbare tools/faciliteiten;
- Bij twijfel ondersteuning inschakelen.

Met name de vanzelfsprekende productkwaliteit is belangrijk maar lastig vast te stellen. De partij die de te leveren productkwaliteit bepaalt, is normaal gesproken het project waarvoor de ontwikkelaars werken. Dit project heeft tenslotte tot doel om een goed werkend product binnen een bepaalde hoeveelheid tijd en geld op te leveren.

Toch is hier een kanttekening bij te plaatsen. Wat doet de ontwikkelaar als een project, in het mastertestplan of ingegeven door tijdsdruk, *geen* expliciete eisen stelt aan de kwaliteit van de opgeleverde software? Of zelfs “in paniek” een oplevering vraagt van de, nog nauwelijks geteste, software? Worden er dan geen ontwikkeltests uitgevoerd? Op het eerste gezicht lijkt het geen probleem om de ontwikkeltests te laten schieten. Als het project de opdracht geeft, mag er ook ongetest geleverd worden … De ervaring leert echter dat het zeer onverstandig is om geen of onvoldoende ontwikkeltests uit te voeren. Hoewel een project de mijlpaal op dat moment haalt, komt er een moment, tijdens de systeem- of acceptatietest of nog erger: in productie, dat de fouten alsnog binnen komen stromen. Uiteindelijk slaat dit in negatieve zin weer terug op de ontwikkelaars.

Om die reden ligt hier ook een verantwoordelijkheid voor de ontwikkelafdeling waar de ontwikkelaars onder vallen. Deze kan voorschrijven dat ongeacht de projectdruk de ontwikkelaars altijd een bewust gekozen basiskwaliteit als minimum moeten leveren.

Wanneer de basiskwaliteit duidelijk wordt vastgelegd, hoeft in de opdrachtformulering voor het ontwikkeltesten binnen een project alleen aandacht te worden besteed aan de delen van het systeem waar het project een hogere zekerheid wenst.

De ontwikkelaar (de ontwikkelafdeling) dient daartoe de diepgang, inzichtelijkheid, bewijsvoering en nalevingcontrole van de ontwikkeltests vastgelegd te hebben. Een belangrijke te maken keuze is de gewenste mate van bewijsvoering van het testen. Hoeveel zekerheid wil men hebben dat de tests inderdaad volledig volgens de afgesproken strategie zijn uitgevoerd? En hoeveel tijd en geld heeft men voor deze bewijsvoering over? Steeds vaker stellen ook externe partijen, bijvoorbeeld toezichthoudende instanties, eisen aan de op te leveren bewijsvoering.

Voorbeeld

Binnen een organisatie wordt basiskwaliteit als volgt gedefinieerd:

Diepgang:

De basisdiepgang van de testdekking is dat alle statements in de gerealiseerde software minimaal 1 keer in een ontwikkeltest zijn geraakt (statement coverage).

Inzichtelijkheid:

Opsomming van te testen situaties met verwijzing naar de ontwikkelbasis (requirements, specificaties, technisch

ontwerp), met aangeven of de vaststelling in de codereview, unittest of unitintegratietest plaatsvindt.

Bewijsvoering:

In de lijst te testen situaties paraferen wat, en door wie, getest is, zonder duidelijk te maken hoe getest is.

Naleving controle:

codereviews (steekproefsgewijs).

Aanvullend op basiskwaliteit kan de ontwikkelafdeling ook kiezen voor een model met meerdere kwaliteitsniveaus. Dit maakt het voor projecten makkelijker om variaties op de te leveren kwaliteit aan te brengen.

Voorbeeld

Een organisatie definieert boven de basiskwaliteit nog drie andere kwaliteitsniveaus.

De basiskwaliteit krijgt het label *brons* en de niveaus daarboven de labels *zilver*, *goud* en *platina* (zie de onderstaande tabel).

	Brons	**Zilver**	**Goud**	**Platina**
Diepgang	Statement coverage	Condition/ decision coverage	Modified condition/ decision coverage	Multiple condition coverage
Inzichtelijkheid	Opsomming van te testen situaties met verwijzing naar de test/ontwikkelbasis (requirements, specificaties, technisch ontwerp), met aangeven of de vaststelling in de codereview, unittest of unitintegratietest plaatsvindt	Testgevallen (logisch), met aangeven of de vaststelling in de codereview, unittest of unitintegratietest plaatsvindt	Testgevallen (logisch en fysiek), met aangeven of de vaststelling in de codereview, unittest of unitintegratietest plaatsvindt	Testgevallen (logisch en fysiek), met aangeven of de vaststelling in de codereview, unittest of unitintegratietest plaatsvindt
Bewijsvoering	Geparafeerde checklists	Testverslagen	Testverslagen + bewijs	Testverslagen + bewijs
Naleving controle	codereviews en interne controle op testresultaten (steekproefsgewijs)	codereviews en interne controle op testresultaten	codereviews, interne controle op testresultaten en steekproef externe audit	codereviews, interne controle op testresultaten en externe audit

Met de creatie van meerdere kwaliteitsniveaus ontstaat een situatie waarbij de opdrachtgever voor de verschillende delen van het systeem de gewenste kwaliteit kiest, en de ontwikkelafdeling per kwaliteitsniveau het prijskaartje er aan hangt. Kortom: een eerste stap naar onderhandelbare *gekozen kwaliteit*. De gekozen kwaliteit is een afspraak over de invulling van de ontwikkeltests met betrekking tot de inzichtelijkheid, de diepgang en de bewijsvoering van de uitgevoerde tests.

Bewijsvoering

Er zijn meerdere mogelijkheden om het vertrouwen te krijgen dat de gewenste kwaliteit ook daadwerkelijk (op tijd) wordt geleverd. Als eerste is er de keuze van bewijsvoering die in de strategiebepaling van de ontwikkeltests is vastgelegd met de zogenaamde exit-/entry-criteria. Deze moeten worden nagekomen voordat de volgende testsoort start. Ook kan de projectmanager aanvullende bewijsvoering verlangen om specifieke projectrisico's te bewaken. De keuze van (aanvullende) bewijsvoering is een afweging tussen ervaringen, risico's en kosten die de bewijsvoering met zich meebrengt.

Mogelijke vormen van bewijsvoering, vaak te combineren, zijn:

- Gemarkeerde/geparafeerde testbasis
  Het in de ontwikkelbasis (requirements, functioneel ontwerp, use case beschrijvingen en/of technisch ontwerp) paraferen van datgene wat getest is, zonder duidelijk te maken hoe getest is.
- Geparafeerde checklist
  Een checklist afleiden van de testbasis (bijvoorbeeld requirements, use case beschrijvingen en/of technisch ontwerp) en paraferen wat getest is, zonder duidelijk te maken hoe getest is.
- Testgevallen
  De met behulp van een bepaalde testontwerptechniek met gekozen diepgang opgestelde testgevallen.
- Testgevallen + testverslagen

Idem aan vorige, plus een verslaglegging van welke testgevallen uitgevoerd zijn met welk resultaat.
- Testgevallen + testverslagen + bewijs
  Idem aan vorige, plus bewijs van de testuitvoering in de vorm van screen- en databasedumps, overzichten, enzovoort.
- Testcoveragetools (tools voor het meten van de bereikte dekkingsgraad)
  De uitvoer van dergelijke tools laat zien wat allemaal getest is, bijvoorbeeld welk percentage van de code of van de interfaces tussen modules is geraakt. Dit kan onderdeel zijn van de build-rapportage.
- Geautomatiseerde tests
  Het geautomatiseerd uitvoeren van tests in de nieuwe omgeving (bijvoorbeeld systeem- of acceptatietestomgeving) toont heel snel aan of de geleverde software dezelfde is als de software die getest is in de ontwikkeltests en of de installatie juist is verlopen.
- Demonstratie
  Demonstreren dat de (deel-)functionaliteit en/of de gekozen architectuur werkt conform gestelde eisen.
- Test-Driven Development naleving rapportage
  Middels reviewverslagen kan aannemelijk worden gemaakt dat de richtlijnen met betrekking tot de toepassing van TDD zijn nagekomen.

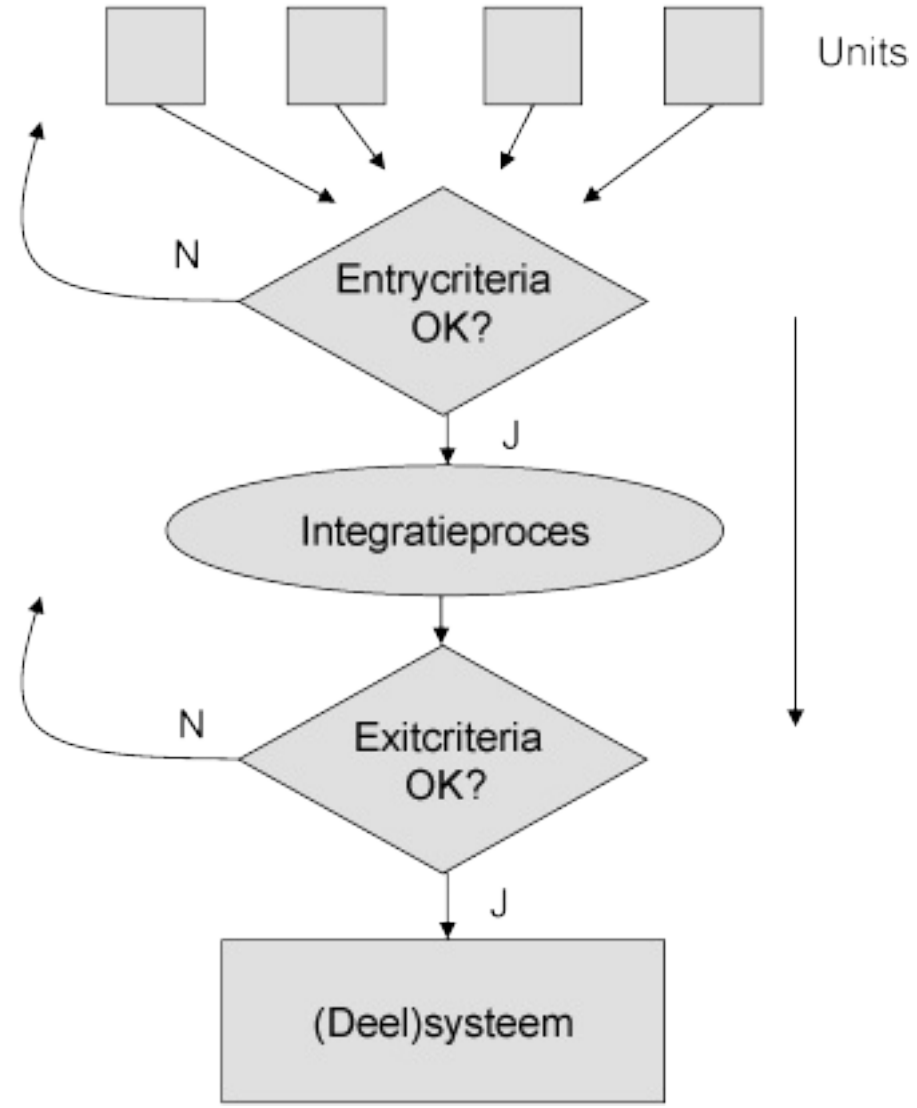
Ook moet afgesproken worden hoe de rapportage eruit komt te zien. Het kan zijn dat een afzonderlijk testrapport wordt opgeleverd, maar het rapport kan ook deel uitmaken van reguliere ontwikkelingsrapportages.

Applicatie-integrator aanpak

Wanneer zowel unittest als unitintegratietest een gestructureerde ontwikkeltestwerkwijze gaan hanteren, ligt het voor de hand om afstemming hier tussen te bewerkstelligen, zodat er geen onnodige overlap of juist gaten in de totale testdekking van de ontwikkeltests zitten. Onderstaand wordt een praktische werkwijze beschreven die deze afstemming vergemakkelijkt, de testverantwoordelijkheden duidelijk belegt en een gemakkelijke opstap biedt voor structurering van het ontwikkeltesten.

In deze aanpak wordt een zogenaamde applicatie-integrator (AI) verantwoordelijk gesteld voor de voortgang van het integratieproces én voor de kwaliteit van het uitgaande product. De AI overlegt met zijn opdrachtgever (de projectmanager of teamleider ontwikkeling) over de op te leveren kwaliteit: onder welke omstandigheden mag het systeem of deelsysteem vrijgegeven worden voor een volgende fase (exit-criteria). Tevens vraagt de AI inzicht in de kwaliteit van de ingaande units (entry-criteria), om vast te stellen of de kwaliteit van deze producten voldoende is om het eigen integratieproces op efficiënte wijze te kunnen doorlopen. Een unit wordt alleen in het integratieproces opgenomen wanneer het aan de entry-criteria voldoet. Een (deel)systeem wordt uitgeleverd indien (aantoonbaar) aan exit-criteria wordt voldaan (zie figuur 7.3). Het mag duidelijk zijn dat het goed hanteren van de exit- en entry-criteria een grote impact heeft op (het inzicht krijgen in) de kwaliteit van de afzonderlijke units en het uiteindelijke systeem. Testen is zeer belangrijk voor het vaststellen van deze criteria, want onderdelen van een criterium zijn bijvoorbeeld het te testen kwaliteitsattribuut, de gewenste dekkingsgraad, het gebruik van bepaalde testontwerptechnieken en het op te leveren bewijsmateriaal. De entry- en exit-criteria worden daarom gebruikt bij de strategiebepaling van de unit- en de unitintegratietest.

Deze werkwijze geldt ook wanneer het integratieproces uit meerdere stappen bestaat of in geval van onderhoud.



Figuur 7.3: Entry- en exit-criteria.

Om belangenverstrengeling te voorkomen heeft de AI bij voorkeur niet tevens de rol van ontwerper of projectleider ontwikkeling. Hiermee wordt bewust een spanningsveld gecreëerd tussen de AI die verantwoordelijk is voor de kwaliteit en de projectleider ontwikkeling die met name wordt beoordeeld op aspecten als de geleverde functionaliteit, doorlooptijd en besteed budget.

De rol van de AI is beschreven in hoofdstuk 16 “Testrollen”.

Opvallende maatregelen in de aanpak zijn:

- Er wordt een bewuste keuze gemaakt voor de op te leveren kwaliteit en de uit te voeren tests vóór oplevering aan een volgende fase (zodat ook inzicht in de kwaliteit verkregen wordt).
- De door de ontwikkelaars uitgevoerde tests worden inzichtelijker.
- Naast de eindverantwoordelijkheid van de project- of teamleider ontwikkeling, is de verantwoordelijkheid voor het testen belegd bij een persoon binnen het ontwikkelteam.

Implementaties van deze aanpak hebben inmiddels aangetoond dat de latere tests een opmerkelijk lager aantal serieuze bevindingen opleveren. Een ander voordeel van de aanpak is dat een eerdere betrokkenheid van de systeemtest en acceptatietest mogelijk is. Omdat er een beter inzicht is in de kwaliteit van de afzonderlijke delen van het systeem, kan er een betere risicoafweging plaatsvinden om bepaalde tests al in een eerder stadium uit te voeren. Een voorbeeld hiervan is dat de acceptatietest al de schermen beoordeelt op gebruikersvriendelijkheid en bruikbaarheid, terwijl de unitintegratietest nog bezig is. Dergelijke tests hebben slechts zin wanneer er al een redelijke mate van vertrouwen is in de kwaliteit van deze schermen en de afhandeling ervan.

7.2.8 Testtools voor ontwikkeltests

In de ontwikkeltests worden andere tools gebruikt dan in de systeem- en acceptatietests. Een aantal van deze soorten tools wordt hieronder besproken. De tools zijn vaak onderdeel van de ontwikkelomgeving. Ook zijn er veel van dit soort tools te vinden op internet. Daarnaast zijn er tools die zowel voor de systeem- en acceptatietests als voor de ontwikkeltests worden gebruikt, zoals code coveragetools, monitoringtools, stubs en drivers. Deze tools staan beschreven in het algemene onderdeel over testtools in paragraaf 8.5.3 “Soorten testtools”. Vermeldenswaardig is wel dat codecoverage-tools veelal zijn geïntegreerd bij ontwikkelomgevingen die geautomatiseerde unittests ondersteunen.

De volgende soorten testtools worden hier toegelicht:

- Debugger;
- Codeanalysetool;
- Unittesttool.

Debugger

Met behulp van een debugger is het bijvoorbeeld mogelijk om de oorzaak van een specifieke fout op te sporen en vervolgens op te lossen. Debuggers maken het mogelijk om, afhankelijk van het hulpmiddel, softwarelogica en -data te bekijken en/of te manipuleren op source- en/of objectniveau.

Codeanalysetool

Er zijn tools die de (source)code als invoer gebruiken en aan de hand daarvan allerlei statische analyses en controles uitvoeren. Het doel is niet zozeer om “harde fouten” op te sporen, maar het signaleren van “onveilig” programmeren en foutgevoelige code. Dit geeft aan de tester bijvoorbeeld informatie over de onderhoudbaarheid van het systeem. Ook wordt deze informatie gebruikt om de complexere en daarmee risicovollere systeemdelen te herkennen. Aan deze delen kan dan relatief meer testinspanning worden gegeven. De moeilijkheid bij dit soort hulpmiddelen is vaak dat deze afhankelijk zijn van de omgeving (hardware, software, enzovoort) waarin ontwikkeld wordt. Het gaat hier om een statische test, in tegenstelling tot een dynamische test waarbij de software daadwerkelijk werkt. Dit betekent dat er geen invoergegevens en geen uitvoervoorspelling nodig zijn en dat de software geen uitvoer genereert. Vaak hebben de tools in de ontwikkelomgeving bepaalde codeanalysemogelijkheden. Zo hebben compilers vaak allerlei ingebouwde controlemogelijkheden. Daarnaast zijn dergelijke tools commercieel verkrijgbaar of worden ze zelf gebouwd. Denk hierbij bijvoorbeeld aan een tool dat de code op (een deel van de) normen en standaards controleert. De functionaliteiten waar statische analysetools zich op richten zijn ruwweg onder te verdelen in drie groepen:

1. *Analyse van de codestructuur*
   Is de structuur van een unit of module op een bepaalde wijze opgezet? De tool tracht bijvoorbeeld een Nassi-Shneiderman diagram te genereren. Als dit niet lukt, vindt signalering van een structuurfout plaats. Door middel van structuuranalyses wordt een beoordeling gegeven van de softwarearchitectuur;
2. *Codeerregels*
   Is ontwikkeld volgens de normen en standaards? Voldoet de code aan de gehanteerde style guide (zijn er bijvoorbeeld voldoende commentaarregels of is op de juiste wijze ingesprongen)? Krijgen alle velden in een unit voordat ze worden gebruikt een waarde? Ontstaan er geen oneindige lussen? Worden de verschillende typen van variabelen niet foutief door elkaar gebruikt? Vaak bieden compilers functionaliteiten om deze vragen te beantwoorden. Zo kunnen compilers naast syntax-controle bijvoorbeeld bepaalde runtime-controles standaard inbouwen en niet-geïnitialiseerde variabelen, ongebruikte code en oneindige loops detecteren. De meeste compilers vervaardigen ook een overzicht van variabelen en het gebruik daarvan, de zogenaamde cross-reference overzichten. Voor webapplicaties zijn HTML controletools beschikbaar.
3. *Metrics van code*
   Met behulp van deze functionaliteit kunnen metrics worden gegenereerd ten aanzien van de code, onder andere in termen van omvang, complexiteit of commentaar frequentie. Een voorbeeld van een complexiteit metric is de formule van McCabe [McCabe, 1976], waarbij een uitspraak wordt gedaan over de mate van complexiteit van de units. Onder complexiteit wordt verstaan het aantal paden dat in een unit doorlopen kan worden. Grondslag van de theorie van McCabe is dat met het toenemen van het aantal beslispunten in een unit de complexiteit van de unit toeneemt en daardoor de kans op fouten.

Unittesttool

Tools voor de unittest zijn specifiek voor de programmataal. Het wordt gebruikt door de ontwikkelaar om testscripts te maken die een unit of een stuk code geautomatiseerd testen in een zogenaamd testraamwerk. De testgevallen zijn als code

dan opgenomen bij de (source)code. Vaak zit er rond een unittesttool een simpele beheerfunctie om meerdere scripts voor de verschillende unittests te beheren. Met de opkomst van test driven development komen unittesttools steeds meer in de belangstelling te staan.

7.3 Testactiviteiten

De ontwikkeltests, dat wil zeggen de testsoorten unittest en unitintegratietest, maken een onlosmakelijk deel uit van de ontwikkelactiviteiten die worden uitgevoerd door de ontwikkelaar. Ze worden niet georganiseerd als een zelfstandig proces met een onafhankelijk team. Ondanks dat kan voor het proces van de unittest en unitintegratietest wel een aantal verschillende activiteiten worden geïdentificeerd. In deze paragraaf worden de activiteiten van de ontwikkeltests beschreven aan de hand van het faseringsmodel van TMap. Hiermee worden de onderlinge volgorde en afhankelijkheden duidelijker en is het beter herkenbaar voor de testadviseur of testmanager. In het TMap faseringsmodel zijn de testactiviteiten verdeeld over een zevental fasen. Deze fasen zijn: Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering, Afronding.

Nadrukkelijk zijn in de praktijk van ontwikkeltesten de activiteiten minder goed zichtbaar en herkenbaar dan in de systeem- of acceptatietest. Ook zijn de resultaten van activiteiten vaak niet goed vastgelegd. Om die reden kan het lezen van onderstaande activiteiten een beeld oproepen van ongewenste formaliteit. Dat is niet zo bedoeld. De beschrijvingen zijn bedoeld om een volledig beeld te geven van de uit te voeren activiteiten, los van de mate van formaliteit van een activiteit.

Tip

De opsomming van activiteiten maakt het moeilijk voor een ontwikkelaar, projectleider, testadviseur of testmanager te bepalen waar te beginnen als het ontwikkeltesten beter ingericht moet worden. Suggesties voor eerste stappen zijn:

- Onderzoek welke testactiviteiten al plaats vinden. Houd er hierbij rekening mee dat de testbegrippen vaak niet (allemaal) bekend zijn bij de ontwikkelaars.
- Breng de discussie met ontwikkelaars/IT-manager/projectmanager op gang over kwaliteit van de opgeleverde software.
- Maak afspraken over te leveren basiskwaliteit (wat testen, hoe testen, benodigde mate van bewijs).
- Ondersteun dit door het gebruik van (lichte) testtechnieken (waar de ontwikkelaars dan in getraind moeten worden).
- Borg de werkwijze door de testrichtlijnen voor basiskwaliteit onderdeel te (laten) maken van de programmeerrichtlijnen (bijvoorbeeld in de ontwikkelreferentiekaart).

Bovenstaande stappen leiden tot een verhoogde kwaliteit van het ontwikkel(test)en dat getypeerd kan worden als basisvakmanschap. Is hogere kwaliteit gewenst, dan kunnen in een vervolgtraject de volgende stappen doorlopen worden:

- Onderzoek de tevredenheid van de ontvangende partij over de inmiddels bereikte kwaliteit van ontwikkeltesten.
- Neem zo nodig maatregelen om fouten te voorkomen (bijvoorbeeld de kwaliteitsmaatregelen uit 7.2.7).
- Onderzoek of geautomatiseerd testen mogelijk is (biedt de ontwikkelomgeving de benodigde faciliteiten) en een nuttige investering is. Implementeer deze werkwijze bij positieve uitkomst.
- Is voor bepaalde onderdelen hogere kwaliteit dan basiskwaliteit nodig? Dan gaan productrisicoanalyse en teststrategie een rol spelen en wordt de bovenstaande discussie over basiskwaliteit gevoerd voor hogere kwaliteitsniveaus.

7.3.1 Fase Planning

De doelstelling van de fase Planning is het formuleren van een samenhangende en gedragen aanpak waarmee de ontwikkeltests goed uitgevoerd kunnen worden. Doordat de ontwikkeltests een onlosmakelijk deel uit maken van de ontwikkelactiviteiten wordt hiervoor geen gescheiden testplan gemaakt. De redenen hiervoor zijn dat een parallel plan organisatorisch alleen maar tot verwarring leidt en dat het ten onrechte de indruk wekt dat het ontwikkeltesten een aparte losstaande activiteit is. Om de betrokkenen toch te kunnen informeren over de aanpak, kan deze in het ontwikkelplan

worden beschreven. Dit wordt dan vaak aangevuld met het distribueren van losse testvoorschriften.

Tip

Bij het beschrijven van de aanpak voor de ontwikkeltests is het aan te raden om nadrukkelijk aandacht te geven aan het *waarom* rond testen. Dan zullen de testsoorten unittest en unitintegratietest een meer vanzelfsprekender plaats krijgen binnen het proces van softwareontwikkeling.

De fase Planning bestaat uit de volgende activiteiten:

1. Vaststellen opdracht;
2. Vaststellen testbasis;
3. Analyseren productrisico's;
4. Bepalen teststrategie;
5. Bepalen begroting;
6. Bepalen planning;
7. Toewijzen testtechnieken;
8. Definiëren testproducten;
9. Definiëren organisatie;
10. Definiëren infrastructuur;
11. Inrichten beheer;
12. Terugkoppelen en fixeren plan.

Vaststellen opdracht

De opdracht voor het ontwikkeltesten wordt in de meeste situaties impliciet gegeven. Bij het verstrekken van een ontwikkelopdracht is het vanzelfsprekend dat ook de ontwikkeltests worden uitgevoerd, maar blijft vaak onderbelicht hóe dit dan moet gebeuren. Om toch de verwachtingen over en weer op elkaar af te stemmen moet er duidelijkheid komen over welk resultaat het project van de ontwikkeltests verlangt. Dat wordt beschreven in een opdrachtformulering en deze moet aansluiten op de opdrachtformulering zoals geformuleerd in het mastertestplan.

De opdrachtformulering bevat de volgende onderwerpen:

- Opdrachtgever
  De opdrachtgever geeft de opdracht tot het uitvoeren van de ontwikkeltest. Vaak is dat de (overall) projectleider of een andere eindverantwoordelijke.
- Opdrachtnemer
  De opdrachtnemer coördineert de ontwikkeltests. Dit kan zijn de projectleider ontwikkeling of een ontwikkelaar, in de rol van testcoördinator of applicatie-integrator.
- Beschouwingsgebied
  Het beschouwingsgebied beschrijft de grenzen van de ontwikkeltests. Wanneer wordt gewerkt volgens het principe van basiskwaliteit en gekozen kwaliteit (zie paragraaf 7.2.7), dan worden hier de delen van het systeem genoemd, waarvoor het aantonen van de basiskwaliteit niet voldoende zekerheid biedt.
- Randvoorwaarden en uitgangspunten
  Randvoorwaarden beschrijven de voorwaarden die derden aan de ontwikkeltests opleggen. Voorbeelden van randvoorwaarden zijn:
  – eisen die gesteld worden vanuit de volgende testsoorten (vaak geformuleerd in entry-criteria);
  – minimaal leveren van basiskwaliteit als eis van de ontwikkelafdeling;
  – aansluiting op gehanteerde ontwikkelaanpak, bijvoorbeeld test driven development of pair programming.
  De uitgangspunten beschrijven de eisen die de ontwikkeltesten stelt aan de anderen.
- Opdracht
  Uiteindelijk worden alle afspraken tussen de ontwikkelaar en opdrachtgever vastgelegd in een opdracht, bijvoorbeeld in het ontwikkelplan. Wanneer er in het mastertestplan van een project geen expliciete eisen met betrekking tot ontwikkeltesten zijn vastgelegd dient deze opdrachtformulering tevens als middel om het niet aanwezig zijn van deze eisen bevestigd te krijgen.

Vaststellen testbasis

Doelstelling van deze activiteit is het vaststellen van de testbasis voor de ontwikkeltests. Op het eerste gezicht lijkt dit geen probleem te leveren. De testbasis is immers gelijk aan de ontwikkelbasis. Toch is het verstandig om stil te staan bij deze activiteit, vooral op het gebied van standaards en richtlijnen kunnen zaken over het hoofd worden gezien. Bij watervalmethoden kan vastgesteld worden of alle documenten in configuratiebeheer zijn opgenomen en zijn goedgekeurd. Bij methoden waar gebruikersparticipatie is vereist, wordt tijdens deze activiteit een lijst aangelegd van de betrokken gebruikers met hun beslissingsbevoegdheid.

Analyseren productrisico's

Aangezien er bij ontwikkeltest nooit sprake is van een onbeperkte hoeveelheid middelen en tijd, is het belangrijk onderbouwd keuzes te maken welke units (of combinaties van units) van het systeem meer of juist minder testinspanning vragen. De basis om hiervoor onderbouwde keuzes in te maken, vormen de risico's van de units van het systeem en ook het systeem als geheel.

Een hulpmiddel bij het bepalen van de risico's is het uitvoeren van een lichte productrisicoanalyse (PRA). In hoofdstuk 9 "Productrisicoanalyse" wordt de PRA beschreven. In een enkel geval voldoet de PRA van het mastertestplan, in andere gevallen wordt een lichte PRA vanuit het ontwikkeltesten geïnitieerd. De manier om de PRA te verlichten is door minder deelnemers uit te nodigen, bijvoorbeeld alleen een paar ontwikkelaars, een gebruiker en de testmanager van het overkoepelende testproces. Voordat de PRA uitgevoerd kan worden, moeten de volgende onderwerpen bekend zijn:

- Wat is de invloed van de gehanteerde ontwikkelmethode en architectuur.
  Deze hebben invloed op de manier waarop de deelobjecten onderverdeeld kunnen worden. Ook brengen deze eigen risico's met zich mee;
- In welke mate wordt gewerkt met geautomatiseerde unit- en integratietests. Dit heeft invloed op de risico's (zie paragraaf 7.2.7);
- Indien er gewerkt wordt volgens het principe van basiskwaliteit en gekozen kwaliteit moet bekend zijn wat de afspraken zijn met betrekking tot deze basiskwaliteit (zie paragraaf 7.2.7).

De risico's die uit de PRA naar voren komen, kunnen ook invloed uitoefenen op de manier waarop het proces van de ontwikkeltests worden ingericht. Ook kan het invloed hebben op de volgorde waarin het systeem ontwikkeld gaat worden. Het kan bijvoorbeeld, gezien de risico's, verstandig zijn om binnen het ontwikkeltraject een Proof Of Concept op te nemen, waarin expliciet een aantal gerichte ontwikkeltests worden uitgevoerd.

Bepalen teststrategie

Op basis van de resultaten uit de PRA wordt bepaald hoe licht of zwaar getest moet worden voor elke unit (of combinaties van units) van het systeem. Dit wordt vastgelegd per testsoort (unittest en unitintegratietest). Per kenmerk/ deelobject-combinatie wordt de testzwaarte bepaald. Indien gewerkt wordt met basiskwaliteit en hogere kwaliteitsniveaus wordt de testzwaarte hieraan gerelateerd. Als verdere detaillering óf als alternatief kan een keuze worden gemaakt uit onderstaande mogelijkheden:

● ● ●	zware test
● ●	gemiddelde test
●	lichte test
S	statisch testen, het controleren en onderzoeken van producten zonder dat software wordt uitgevoerd

Als in een cel niets staat, betekent dit dat de betreffende toets- of testsoort geen aandacht hoeft te besteden aan het kenmerk. Het resultaat van de activiteit "bepalen teststrategie" wordt vastgelegd in een strategiematrix en kan er als volgt uit zien:

Voorbeeld Unittest

Kenmerk	RK MTP	UNIT MTP		Routines schermvalidaties	Databaseroutines	Units afrekenmodule
Functionaliteit	B	● ● ●		A/ ● ● ●	B/ ● ●	A/ ● ● ●
Performance online	B	●		-	A/ ● ● ●	C/ ●
…						

RK MTP	=	Vanuit het mastertestplan toegekende risicoklasse aan het kenmerk
UNIT MTP	=	Vanuit het mastertestplan toegekende testzwaarte aan de test (in dit geval de unittest)
A, B, C	=	Vanuit de PRA toegekende risicoklasse, A=Hoog, B=Gemiddeld, C= Laag

Daarnaast moet overwogen worden of codereviews noodzakelijk zijn, hetzij op alle code, hetzij op de code van de risicovollere units, hetzij op de middels parameters geconfigureerde componenten en/of tools.

Bepalen begroting

Op basis van de gekozen testaanpak en -zwaarte dient bekeken te worden of de begroting van de ontwikkeling bijgesteld dient te worden. Vaak is de begroting al standaard opgenomen in het ontwikkelbudget. Alleen wanneer de eisen scherper staan dan de basiskwaliteit, zal er een opslag voor de extra testactiviteiten begroot moeten worden. Een ander aandachtspunt is of bij automatisering van de unittests de hiervoor benodigde extra inspanning wel is begroot.

De begroting van de unitintegratietest zal veel meer bepaald worden door de omvang en gekozen strategie. De verschillende begrotingstechnieken en de stappen om tot een begroting te komen staan beschreven in hoofdstuk 11 “Begrotingstechnieken”.

Bepalen planning

De ontwikkelaars werken met één planning: de ontwikkelplanning. Deze ontwikkelplanning dient gecheckt te worden of er ruimte is voor de uit te voeren ontwikkeltests. Unittest- en integratie(test)activiteiten zijn meestal impliciet al opgenomen, maar de vraag is of daar voldoende tijd voor gereserveerd is. Activiteiten voor de testvoorbereiding, codereviews, herstel en hertest worden snel vergeten.

Wanneer blijkt dat de planning geen ruimte biedt voor de ontwikkeltests, is dit het moment om af te stemmen met de opdrachtgever. Zijn de opgestelde teststrategie en hieruit volgende begroting en planning niet acceptabel, dan herhalen deze stappen zich. Hierbij kan de afweging worden gemaakt bepaalde units minder zwaar te testen (maar minimaal met basiskwaliteit). Hierdoor kan tijd en/of geld bespaard worden, maar wordt er meer risico gelopen.

Wanneer de strategie wordt aangepast ontstaat een nieuwe versie van de strategiematrix. Een aangepaste strategiematrix ziet er als volgt uit, waarbij minder testzwaarte wordt getoond door ○ in plaats van ● en meer testzwaarte door ◉.

Voorbeeld Unittest

Kenmerk	RK MTP	UNIT MTP		Routines schermvalidaties	Databaseroutines	Units afrekenmodule
Functionaliteit	B	● ● ●		A/ ● ● ○	B/ ● ● ◉	A/ ● ● ●
Performance online	B	●		-	A/ ● ● ○	C/ ●
…						

Toewijzen testtechnieken

Nadat de definitieve teststrategie is vastgesteld, die aansluit op de begroting en planning van het ontwikkelproces, moeten de ontwikkeltests verder geconcretiseerd worden. Dit gebeurt door middel van het toewijzen van testtechnieken aan de verschillende (groepen van) units. Welke testtechniek gebruikt gaat worden is afhankelijk van de gekozen testzwaarte. Geschikte testontwerptechnieken voor ontwikkeltesten zijn bijvoorbeeld de elementaire vergelijkingentest en de beslistabeltest. Technieken als error guessing of checklists zijn eveneens goed toepasbaar.

De aanpak van de integratie van een systeem gebeurt normaal gesproken in een aantal stappen waarbij telkens één of meer units met elkaar worden geintegreerd en getest.

Voor de unitintegratietest zijn er verschillende mogelijkheden voor de volgorde van unitintegratie:

- Top-down
  Er wordt van boven naar beneden getest (bijvoorbeeld werkend vanuit het menuscherm). Units worden vervangen door stubs (zie paragraaf 8.5 “Testtools”);
- Bottom-up

Van onder naar boven wordt getest (bijvoorbeeld eerst datamanipulatie en later pas het bijbehorende scherm). Units worden vervangen door drivers (zie paragraaf 8.5 “Testtools”);
- Beschikbare delen eerst
  Integreren puur op basis van de oplevervolgorde van de units;
- Functie voor functie
  Integreren op basis van de functies, zoals vastgelegd in het functioneel ontwerp.

Wat de beste integratievolgorde is en hoeveel integratiestappen noodzakelijk zijn, is afhankelijk van de plaats van de meest risicovolle delen van het systeem. Bij de integratietest zijn, afhankelijk van de omvang van te integreren onderdelen, vooral technieken op basis van paden of equivalentieklassen aan te bevelen.

Definiëren testproducten

Het doel van deze activiteit is het eenduidig definiëren van de op te leveren testproducten en van de wijze waarop gerapporteerd wordt. Op basis van de teststrategie zijn testtechnieken gekozen. Bovendien kunnen afhankelijk van de afgesproken kwaliteit bepaalde bewijsmaterialen van de test nodig zijn.

Gebruik hiervan resulteert in de volgende testware:

- testgevallen, checklists, enzovoort;
- testresultaten in de vorm van ingevulde checklists, logging-verslagen, screendumps, afgevinkte verslagen, enzovoort.

Definiëren organisatie

Ook ontwikkeltesten heeft organisatie nodig. De primaire testtaken liggen bij de ontwikkelaars, maar de verantwoordelijkheid voor het testproces moet belegd worden bij één persoon. Dit kan iemand zijn met de rol testcoördinator maar dit kan ook de eindverantwoordelijke voor de ontwikkeling zijn. Daarnaast dient bekend te zijn waar ondersteuning op het gebied van testkennis gehaald kan worden.

In geval van handmatig uit te voeren unittests dient de keuze gemaakt te worden of de unittest wordt gedaan door de maker van de unit, door een collegaontwikkelaar of, bij pair programming, door een duo. Het voordeel van het zelf testen is, dat de ontwikkelaar veel kennis heeft van de interne werking van de unit. Dit is het meest efficiënt omdat er geen inwerktijd is en vrijwel geen overhead aan communicatie. Het voordeel van het testen door een collega-ontwikkelaar is dat mogelijke “blinde vlekken” van de ontwikkelaar eerder worden ontdekt en dat de collega een objectievere test uitvoert dan de ontwikkelaar zelf. Een variant hierop is dat de ontwikkelaar de test van de eigen unit uitvoert, maar dat een collega-ontwikkelaar deze test voorbereidt. Bovengenoemde te maken keuzes hangen overigens sterk samen met de teststrategie.

Voor de unitintegratietest kan de verantwoordelijkheid voor het testproces belegd worden bij de applicatie-integrator, zie paragraaf 7.2.7. Deze applicatie-integrator is verantwoordelijk voor de voortgang van het integratieproces én voor de kwaliteit van het uitgaande product. Omdat de applicatie-integrator niet de veelheid aan taken en verantwoordelijkheden van de projectmanager heeft, krijgt het testen, en daarmee de kwaliteit van het testobject, meer aandacht. De rol applicatie-integrator wordt verder toegelicht in hoofdstuk 16 “Testrollen”.

Bij ontwikkeltesten is een grondige systeemkennis met voldoende testkennis vereist. In de praktijk blijkt de systeemkennis veelal voldoende aanwezig te zijn, maar de testkennis vaak te ontbreken. Manieren om deze kennis op peil te brengen zijn:

- opleidingen, op het gebied van zowel testtechnieken en -management als van testbewustwording;
- ondersteuning en coaching door een testspecialist.

Definiëren infrastructuur

Hoewel ontwikkeltesten normaal gesproken een onderdeel is van het totale ontwikkelingsproces, kunnen er vanuit het testen specifieke eisen zijn ten aanzien van de infrastructuur. Het definiëren hiervan heeft tot doel deze eisen in een zo vroeg mogelijk stadium helder te krijgen, zodat passende maatregelen genomen kunnen worden. Bij specifieke eisen moet bijvoorbeeld worden gedacht aan koppelingen met aanpalende systemen, autorisaties om te kunnen monitoren, het benodigde aantal testdatabases of het kunnen manipuleren van de systeemparameters. Ook eventueel benodigde testtools

worden hier geïdentificeerd. Voor meer informatie over testtools die bij de ontwikkeltests gebruikt kunnen worden, zie paragraaf 7.2.8 en paragraaf 8.5.3 “Soorten testtools”.

Inrichten beheer

Onder beheer wordt verstaan het beheer van het ontwikkeltesten, de infrastructuur en de testproducten. Dit maakt meestal onderdeel uit van de normale ontwikkelactiviteiten. Het enige dat afwijkt is het bevindingenbeheer. In de systeemtest en acceptatietest is het administreren van elke gedane bevinding “heilig”, want de inhoud van deze administratie wordt gebruikt om inzicht te krijgen in de voortgang en nog uit te voeren activiteiten (hertests), om rapportages op te stellen, voor het verzamelen van metrics en om trends te analyseren.

Voor de ontwikkeltests is dit weliswaar ook mogelijk, maar veel minder noodzakelijk. Wanneer een unittester een bevinding doet, maar deze bevinding gelijk wordt opgelost en gehertest, dan hoeft de bevinding niet te worden geadministreerd (tenzij de unit al elders gebruikt wordt). Op deze manier wordt voorkomen dat een ontwikkelaar enerzijds “de vuile was moet buitenhangen” en anderzijds veel tijd kwijt is aan administratie. Bij twijfel inzake de oorzaak of onduidelijkheid over de ontwikkelbasis, worden de bevindingen wel uitgeschreven, want in dat geval moeten ze worden voorgelegd aan de ontwerper binnen het ontwikkelteam. Vanaf het moment dat het testobject formeel in beheer genomen wordt en elders gebruikt gaat worden, dienen de al bekende “known errors” en vanaf dan te vinden fouten geadministreerd te worden.

Terugkoppeling en fixeren plan

De laatste activiteit van de fase Planning betreft het vastleggen van de resultaten van alle tot nu toe uitgevoerde activiteiten en het verkrijgen van goedkeuring van de gekozen aanpak door de opdrachtgever. De resultaten van alle tot nu toe uitgevoerde activiteiten worden vastgelegd in het ontwikkelplan. Deze resultaten moeten consistent zijn met elkaar. In de praktijk verloopt het opstellen van een consistent plan in een aantal slagen.

De resultaten worden teruggekoppeld met de opdrachtgever en andere betrokken partijen (zoals de testmanager van het overkoepelende testproces) voor goedkeuring of bijstelling. Dit maakt de te volgen testaanpak transparant en stuurbaar, geheel in lijn met BDTM. In het geval van bijstelling kan dat resulteren in een aangepaste strategie, waarbij het schrappen of toevoegen van testzwaarte wordt getoond door respectievelijk ◌ of ● in plaats van ●, zoals al eerder aangegeven bij de activiteit “Bepalen planning”.

7.3.2 Fase Beheer

Het doel van de fase Beheer is om de opdrachtgever voor de ontwikkeltests voldoende inzicht in én sturingsmogelijkheden te geven over:

- de voortgang van de ontwikkeltests;
- de kwaliteit en risico’s van de units (of combinaties van units)
- de kwaliteit van het ontwikkeltesten.

Hiervoor dient de verantwoordelijke de ontwikkeltests op optimale wijze te beheersen en hierover te rapporteren. De fase Beheer bestaat uit de volgende activiteiten:

1. Uitvoeren beheer;
2. Bewaken;
3. Rapporteren;
4. Bijsturen.

Uitvoeren beheer

Doel van het uitvoeren van beheer is om voortdurend inzicht te kunnen bieden in de voortgang en kwaliteit van het ontwikkeltesten en de kwaliteit en risico’s van de units (of combinaties van units). Naast het uitvoeren van de beschreven procedures wordt ook het ontwikkeltesten ondersteund met, en gecontroleerd op, het toepassen van normen en standaards voor beheer. De echte uitdaging voor de testcoördinator of applicatie-integrator bij het beheer is niet zozeer het zelf volgen van de procedures, maar het zorgen dat de ontwikkelaars dat ook doen.

Bewaken

Het bewaken van de uitvoering van het plan is de moeilijkste taak. Het is meer een communicatieve activiteit dan een instrumentele activiteit. Het omvat enerzijds het bewaken van de activiteiten, anderzijds het bewaken van de wereld rondom het ontwikkeltesten. Zijn er ontwikkelingen die hierop van invloed zijn? Of misschien ontbrak bepaalde informatie bij het opstellen van het plan. Zo kan bijvoorbeeld de situatie bestaan dat, op basis van de door codeanalysetools (zie paragraaf 7.2.8) gemeten complexiteit van units, wordt besloten om detailaanpassingen op de strategie en testzwaarte te maken.

Rapporteren

Periodiek en op verzoek wordt, conform de afspraken die in de opdrachtformulering zijn vastgelegd, gerapporteerd over de voortgang van het (test)proces en de kwaliteit van het testobject. De voortgangs- en risicorapportage worden veelal gecombineerd tot één rapport dat opgenomen wordt in de al op te stellen voortgangsrapportages binnen het systeemontwikkelingsproces. De inhoud van rapportage is afhankelijk van de gedefinieerde basiskwaliteit en de gekozen kwaliteit. In de rapportage worden bijvoorbeeld de volgende aspecten schriftelijk vastgelegd:

- Status van het testobject (per unit, groep van units of integratiestap)
- totaal aantal testgevallen;
- aantal testgevallen nog uit te voeren;
- aantal testgevallen correct uitgevoerd;
- Knelpunten en bespreekpunten.

En aan het eind van een test:

- evaluatie van het testobject, wat is er getest, inclusief zogenaamde known errors, d.w.z. bevindingen die (nog) niet opgelost zijn;
- evaluatie teststrategie, waarbij eventuele afwijkingen van de oorspron kelijk afgesproken teststrategie worden gecommuniceerd naar de opdrachtgever.

Bijsturen

Wanneer een voorgestelde maatregel uit de rapportage moeten worden doorgevoerd dan gebeurt dat binnen deze activiteit. Maatregelen kunnen zijn:

- Aanpassen van de teststrategie (zie activiteit ‘Bewaken’);
- Aanpassen van het beschouwingsgebied. Dit kan bijvoorbeeld het geval zijn wanneer ook de scope van ontwikkeling wordt aangepast;
- Aanpassen organisatie. Dit kan bijvoorbeeld het geval wanneer blijkt dat de gekozen structuur niet werkt in de praktijk of wanneer er opleidingen nodig blijken te zijn.

7.3.3 Fase Inrichting en beheer infrastructuur

Er kunnen vanuit het testen specifieke eisen zijn ten aanzien van de infrastructuur. Wanneer dat het geval is, dan wordt dat binnen de fase “Inrichting en beheer infrastructuur” verder ingevuld. Wanneer er geen specifieke eisen zijn wordt deze fase niet ingevuld.

De specifieke eisen kunnen betrekking hebben op de volgende onderdelen van de infrastructuur:

- De omgeving waarin getest wordt;
- De tools die gebruikt worden voor de test;
- De werkplek van de ontwikkelaar.

De eisen kunnen worden gesteld aan de inrichting, gebruik en beheer van deze onderdelen. Ook kunnen eisen worden gesteld aan het later conserveren van (onderdelen van) de infrastructuur voor hergebruik.

Tip

Omgevingsdata als specifieke eis

Iets wat vaak vergeten wordt is de omgevingsdata voor de omgeving waarin getest wordt. De omgevingsdata is de set aan gegevens die de omgeving nodig heeft om te kunnen werken (gebruikersprofielen, netwerkadressen, stamtabellen enzovoort). Om nu tijdens de ontwikkeltests al een voldoende representatieve situatie te simuleren is het aan te raden hier aandacht aan te besteden. Dit betekent niet dat er productiegegevens gebruikt moeten worden, maar wel gegevens die zoveel mogelijk voldoen aan de eisen en richtlijnen die gesteld worden vanuit productie. Bijvoorbeeld een codering voor een gebruiker moet dezelfde structuur hebben als in productie. Hierbij moet de afweging gemaakt worden tussen gewenste flexibiliteit van de ontwikkel(test)omgeving en gewenste representativiteit.

7.3.4 Fase Voorbereiding

Hoewel in de fase Planning de (definitie van de) testbasis is vastgelegd, is op het moment van het uitvoeren van deze fase de testbasis vaak nog niet beschikbaar. In de fase Voorbereiding moet worden onderzocht of de opgeleverde testbasis overeenstemt met en bruikbaar is voor de eerder vastgelegde afspraken in het plan. Als dit niet het geval blijkt te zijn kan het nodig zijn het plan ‘bij te sturen’. Het onderzoeken van de testbasis gebeurt door middel van de uitvoering van een detailintake.

Invulling van de intake is sterk afhankelijk van de gehanteerde ontwikkelmethode. Bij iteratieve en agile methoden waar gebruikersparticipatie een belangrijke rol speelt, valt er over het algemeen niet meer te doen dan het intaken van de te hanteren richtlijnen en duidelijkheid krijgen over de beslissingsbevoegdheid van de betrokken gebruikers. Bij watervalmethoden waar documentatie belangrijk is, vormen de opgeleverde specificaties vaak een ontwikkelcontract. De ontwikkeltestintake wordt dan onderdeel van de ontwikkelintake. Het doel van de ontwikkelintake is het onderzoeken of de opgeleverde specificaties voldoende bouw- en testbaar zijn voor een beheerste voortgang.

De ontwikkelintake omvat de toetsactiviteiten die duidelijkheid geven over de volledigheid, eenduidigheid en consistentie van de functionele specificaties, waardoor een uitspraak kan worden gedaan over de ontwikkelbaarheid, de testbaarheid en/of de condities waaronder een realisatieverplichting kan worden aangegaan.

7.3.5 Fase Specificatie

In de fase Specificatie specificeren de testers per unit respectievelijk integratiestap de tests voor zover nodig aan de hand van de toegewezen testtechnieken uit de testbasis (ontwikkelbasis) en maken ze eventuele hulpprogrammatuur voor het testen met als doel zoveel mogelijk voorbereid te hebben om de testuitvoering zo snel mogelijk te laten verlopen.

Binnen de fase Specificatie worden de volgende activiteiten onderkend:

1. Onderzoek testpatterns (optioneel);
2. Opstellen testspecificaties;
3. Vervaardigen hulpprogrammatuur.

Onderzoek test patterns (optioneel)

Net als bij het ontwerpen van functionele code zijn veel problemen bij het ontwerpen en coderen van de tests al eerder opgelost. Het is dus belangrijk niet steeds opnieuw het wiel uit te vinden! Sinds de introductie van objectoriëntatie gebruiken ontwikkelaars design patterns om ontwerpvraagstukken binnen hun projecten op te kunnen lossen. Patterns geven een formele benadering om een ontwikkelprobleem te omschrijven, met de voorgestelde oplossing en de daarbij horende factoren die effect kunnen hebben op het probleem of de oplossing. Met name bij iteratieve en agile methoden hebben met de opkomst van geautomatiseerde ontwikkeltests ook de *test patterns* hun intrede gedaan.

Definitie

Een *test pattern* is een algemene oplossing voor een specifiek terugkomend testprobleem.

De testwereld heeft inmiddels een aantal bruikbare test patterns. In deze optionele activiteit onderzoeken de ontwikkeltesters welke test patterns van toepassing kunnen zijn op de in te bouwen tests. Hoewel ontstaan bij het geautomatiseerd testen, zijn test patterns ook heel goed toepasbaar bij het ontwerp van testgevallen voor handmatige tests.

Uitgediept

Test patterns worden in de literatuur beschreven met de volgende informatie: naam, doel, context, foutmodel (omgevingsfactoren; voorwaarden voor detectie), voorgestelde oplossing, entry-criteria, exit-criteria, gevolgen, bekend gebruik en gerelateerde patterns.

Voorbeelden van beschikbare test patterns zijn:

- pass/fail pattern,
- collection management pattern,
- data driven pattern,
- performance pattern,
- process pattern,
- simulation pattern,
- multi threading pattern,
- stress test pattern.

Voor meer informatie wordt verwezen naar [Clifton, 2004]

Opstellen testspecificaties

De creatie van testge vallen vindt plaats per unit, respectievelijk integratiestap aan de hand van de toegewezen testtechnieken, waarbij het ook mogelijk is dat voor een bepaalde unit geen techniek is voorgeschreven. Wanneer geen techniek is voorgeschreven, heeft de ontwikkelaar de verantwoordelijkheid om tijdens de ontwikkeling tegelijkertijd de testgevallen te ontwerpen én uit te voeren, waarbij gedocumenteerd wordt conform de afspraken in de strategiebepaling en conform de afgesproken kwaliteit.

Uitgediept

Voor de situatie dat met een testraamwerk (geautomatiseerde unit- en unitintegratietests) wordt gewerkt, worden de eisen vanuit de teststrategie, voorzover ze scherper zijn dan de basiskwaliteit, meestal in het technisch ontwerp gespecificeerd, zodat de ontwikkelaar dit niet kan vergeten tijdens het coderen van de tests.

Het niet voorschrijven van een techniek betekent niet dat het testen overgeslagen mag worden, de basiskwaliteit (100% statement coverage in het voorbeeld van paragraaf 7.2.7) moet altijd worden gehaald.

Het is mogelijk dat er gedurende deze activiteit tekortkomingen en/of onduidelijkheden worden geconstateerd in de ontwikkelbasis. Het is uiteraard van belang deze zo snel mogelijk vast te leggen, opdat deze zo spoedig mogelijk kunnen worden verbeterd respectievelijk verduidelijkt. De registratie en aanmelding van deze bevindingen dient plaats te vinden door middel van de algemene in het ontwikkelingsproces vastgestelde procedures.

Ten opzichte van de uitvoeringsfase neemt deze activiteit minder uren in beslag dan bij systeem- en acceptatietesten, omdat de testgevallen alleen worden uitgeschreven als de ontwikkelbasis niet rechtstreeks geschikt is als basis om te testen, tenzij er in de strategiebepaling expliciete eisen voor zijn vastgelegd.

Vervaardigen hulpprogrammatuur

Om de test van een unit of een integratiestap goed te kunnen uitvoeren, is het in een aantal gevallen noodzakelijk stubs

(ook wel mock-objects genoemd) of drivers te vervaardigen. De stubs vervangen units die door de te testen units worden aangeroepen. De drivers vervangen units die de te testen units aanroepen.

De communicatie tussen de te testen units en de stubs en drivers geschiedt door middel van een interface. De stubs en drivers moeten zodanig zijn dat deze interfaces reële waarden bevatten en in overeenstemming zijn met de uiteindelijke, werkelijke interfaces tussen de diverse units. Uiteraard komen ook de stubs en drivers zelf ook voor testen in aanmerking.

7.3.6 Fase Uitvoering

De fase uitvoering van de ontwikkeltests gebeurt parallel aan het ontwikkelen en integreren zelf. Ingeval van Test Driven Development start het daadwerkelijk ontwikkelen met het coderen van de unittests (zie paragraaf 7.2.7). Ook als niet met een dergelijke aanpak wordt gewerkt, verdient het de aanbeveling om het coderen en testen cyclisch uit te voeren. In één keer intikken van alle code leidt, achteraf, over het algemeen, tot een lang zoeken naar de oorzaak van fouten. Bij dergelijke herstelslagen raakt ook het overzicht of alle statements bij de test minimaal één keer zijn geraakt snel verloren.

De activiteiten die in het kader van ontwikkeltesten kunnen worden onderscheiden zijn:

1. Ontwikkelen unittests (TDD)
2. Uitvoeren (her)tests;
3. Controleren en beoordelen testresultaten.
4. (Laten) uitvoeren codereview (optioneel);

Ontwikkelen unittests (TDD)

Indien sprake is van Test Driven Development:

- Implementeer de unittest voordat gestart wordt met codering van de functionaliteit;
- Alle codepaden moeten worden getest;
- Voeg eventuele extra testdekking toe op basis van het technisch ontwerp (bij afgesproken kwaliteit);
- Maak de unittest onafhankelijk van andere tests, zodat fout lopen van een testgeval geen gevolgen heeft voor andere testgevallen;
- In het geval van een bevinding, schrijf voor het oplossen eerst een unittest die de aanwezigheid van de bevinding aantoont (en na het oplossen de afwezigheid ervan).

Uitvoeren (her)tests

Uitermate belangrijk gedurende de uitvoering van de tests is discipline. De tests (unittests en integratietests) dienen te worden uitgevoerd zoals bepaald in de teststrategie. Als hiervan wordt afgeweken, dient dit altijd gerapporteerd te worden aan de projectleider ontwikkeling, testcoördinator of applicatie-integrator, zodat maatregelen genomen kunnen worden. Als alle tests met goed gevolg zijn doorlopen is de unit of de integratie klaar. Het heeft geen nut om op eigen initiatief extra tests te gaan uitvoeren. Dit maakt het ontwikkelen (onnodig) duurder. Voer de testdekking uit conform de gemaakte afspraken: niet meer, niet minder!

Controleren en beoordelen testresultaten

De testresultaten worden vergeleken met de verwachte resultaten. Bij verschillen kan de oorzaak gelegen zijn in een ontwikkelfout. Als de unittest door de ontwikkelaar zelf wordt uitgevoerd, worden ontwikkelfouten onmiddellijk hersteld, totdat alle testgevallen goed verlopen. Er zijn echter ook andere mogelijke oorzaken: onduidelijkheden in de ontwikkelbasis, fouten in de ontwikkelomgeving, maar ook fouten in de testgevallen. Alle problemen buiten de directe verantwoordelijkheid van de ontwikkelaar en in bepaalde situaties ook de ontwikkelfouten worden formeel aangemeld volgens de vastgestelde procedures. Na herstel van een bevinding worden de desbetreffende tests opnieuw uitgevoerd tot alle tests zijn uitgevoerd en er geen openstaande bevindingen meer aanwezig zijn.

De fouten die niet opgelost (kunnen) worden, worden samengevoegd tot een known-errors testverslag.

(Laten) uitvoeren codereview (optioneel)

Tijdens, of vlak na, het uitvoeren van de unittest kan een codereview uitgevoerd worden. Door de (aantoonbare) uitvoer van een codereview te benoemen als exit-criterium van de unittest, is bij de start van de integratie al vastgesteld dat conform de ontwikkelrichtlijnen en -standaards is gewerkt.

7.3.7 Fase Afronding

Het doel van de fase Afronding is om de ontwikkeltest op een zodanige manier te beëindigen dat een volgende testsoort óf een volgende ontwikkeltest efficiënter en effectiever kan verlopen.

De fase Afronding kent de volgende activiteiten:

1. Opleveren aan de volgende testsoort;
2. Conserveren testware;
3. Evalueren testproces (optioneel).

Opleveren aan de volgende testsoort

Opleveren aan een volgende testsoort (meestal de systeemtest) kan op vele manieren. De meest kale variant, het opleveren van installeerbare software, heeft als groot nadeel dat vooral in de beginfase van de volgende testsoort veel schijnbaar blokkerende fouten worden aangemeld, die uiteindelijk vooral te maken hebben met onbekendheid met de bediening en met verkeerde autorisatie- of parameterinstellingen.

De meest ideale oplevering is een demonstratie waarbij de ontwikkelaar (vaak de applicatie-integrator) demonstreert dat het systeem in de (volgende) testomgeving functioneert. Dit kan bijvoorbeeld aan de hand van een afgesproken kleine set functionele testgevallen.

Conserveren testware

Met deze activiteit wordt bepaald welke testware bewaard moet blijven voor toekomstige (her)tests, zodat deze tests met minimale aanpassing kunnen worden uitgevoerd. Met name de geautomatiseerde tests zijn van groot belang om te conserveren, omdat in de opvolgende tests de investering zich echt begint terug te verdienen. Indien mogelijk in overleg met de toekomstige beheerder van het systeem (en anders de opdrachtgever) wordt geïnventariseerd welke documentatie, tools of code beschikbaar wordt gesteld. De op te leveren testproducten, waaronder het bewijsmateriaal bij afgesproken kwaliteit, worden vastgelegd. Als laatste vindt de daadwerkelijke overdracht van de testware plaats.

Evalueren testproces (optioneel)

Aan het eind van de test wordt vastgesteld hoe het testproces is verlopen. Van belang hierbij is met name de evaluatie van de teststrategie en de gehanteerde methoden en technieken en de terugkoppeling van de resultaten uit de volgende testsoorten (vooral de bevindingen die tijdens de ontwikkeltests gevonden hadden moeten worden).

8 Ondersteunende processen

8.1 Inleiding

In de vorige hoofdstukken is gesproken over het testproces van TMap en hoe dit georganiseerd wordt. In de praktijk wordt het testen vaak uitgevoerd in projecten. Naast deze projectmatige aanpak is het ook mogelijk het testen (of delen daarvan) te organiseren vanuit een lijnorganisatie. Steeds meer organisaties kiezen hier voor en dit kan verklaard worden uit vier de ontwikkelingen die kernachtig worden samengevat als "*meer voor minder en sneller en beter*" (zie paragraaf 1.2 "TMap evolueert mee").

Deze ontwikkelingen hebben een impact op alle disciplines binnen de IT. Zo ontstaan er nieuwe (zeer geavanceerde) ontwikkelomgevingen voor programmeurs waarmee relatief eenvoudig en snel complexe programma's ontworpen en gebouwd kunnen worden. Ook ontstaan er nieuwe (iteratieve) ontwikkelmethoden die uitgaan van verregaande interactie met de gebruikers waardoor onder andere de doorlooptijd van projecten vele malen korter wordt. Er wordt daarnaast steeds meer gestuurd op hergebruik van (interne en externe) componenten die geïntegreerd moeten worden in de bestaande IT-architecturen.

Kortom, er wordt tegenwoordig in dezelfde hoeveelheid tijd meer gebouwd en onderhouden in de IT dan vroeger. Dat zorgt voor meer testwerk. Daarnaast is, door het uitgebreider en complexer worden van de IT, het bedenken van de juiste testaanpak voor deze IT veel moeilijker geworden. Dit in combinatie met de toenemende aandacht voor kwaliteit binnen organisaties zorgt er voor dat het relatieve aandeel van testen binnen de IT toeneemt.

Gevolg is vaak dat door het management het testen met alle bijbehorende middelen duur gevonden wordt. Andere meningen zijn dat het te lang duurt en dat de kwaliteit van het testproces te wensen overlaat. Nu kent het proces van TMap een aantal vaste proceselementen die bij iedere test terugkomen. Denk bijvoorbeeld aan de concrete activiteiten rond het opzetten van een testorganisatie, testomgeving of het inzetten van testautomatisering. Maar ook de activiteiten als het opdoen van testervaring, leren van het testobject of het definiëren van gebruikte standaarden komen in elk testtraject terug. In een projectmatige testaanpak worden deze elementen alleen ingericht voor de duur van het project. Bovendien wordt met projectstandaarden gewerkt die vaak bij aanvang bedacht worden. Dit geeft opstartkosten. Wanneer er wordt gewerkt met tijdelijk personeel, zal het leereffect beperkt zijn. Aan het einde van het project raakt de opgedane kennis en ervaring grotendeels verloren. En de met moeite tot stand gekomen testomgeving wordt ontmanteld. Verder moet een project ook binnen budget en voorop gezette planning afgerond worden. Hierdoor is er weinig aandacht voor het uitvoeren van activiteiten waarvan de inspanningen niet binnen het project terugverdiend kunnen worden. Het opzetten van testware voor hergebruik is een activiteit, die niet of nauwelijks wordt uitgevoerd, hoewel dit door een gestructureerde testaanpak als TMap wel wordt geëist.

Dit alles is bepaald geen optimale situatie waar het gaat om beheersing van kosten, tijd en kwaliteit. Een oplossing hiervoor is om de uitvoering van deze vaste proceselementen van TMap in een lijnorganisatie te beleggen. Deze lijnorganisatie noemen we dan een permanente testorganisatie. Hierdoor blijft de kennis rond deze proceselementen en de bijbehorende producten behouden en kunnen er activiteiten ontwikkeld worden met langere terugverdientijd. Wanneer projecten dan gebruik willen maken van een van deze proceselementen (zoals een testomgeving, het inzetten van testtools of gebruik van standaardsjablonen) kunnen deze worden afgenomen als dienst van de permanente testorganisatie.

8.2 Testbeleid

Welke elementen van het testproces worden belegd in de permanente testorganisatie en hoe deze ingericht wordt hangt af van het testbeleid dat wordt gekozen binnen de organisatie.

Definitie

Het testbeleid beschrijft hoe een organisatie omgaat met de mensen, middelen en methoden rondom het testproces in de verschillende situaties.

Het testbeleid moet gelden voor alle typen systemen, infrastructuren en ontwikkelmethoden. Omdat testen één van de instrumenten is om kwaliteit te waarborgen zal het testbeleid moeten aansluiten op de andere beleidsmaatregelen en

initiatieven betreffende kwaliteitsmanagement. Het is aan te raden het testbeleid te laten aansluiten op het strategisch, tactisch en operationeel beleid van de organisatie. Op strategisch niveau moet bepaald worden wat de invloed is van het organisatiebeleid ten aanzien van het testen voor de gehele organisatie. Dit vormt het strategisch testbeleid en dat moet vanaf dit niveau voorgeschreven en actief ondersteund worden. Op tactisch niveau moet dit testbeleid vertaald worden naar de invulling per organisatieonderdeel, afdeling, productgroep, programma of project (afhankelijk van de inrichting van de betreffende organisatie). Dit betreft ook de resources en budgetten om onvoorwaardelijk toepassing van het testbeleid te garanderen. Het consequent implementeren van dit testbeleid leidt tot een uniforme testaanpak op operationeel niveau.

Strategisch

Het strategische beleid van een (IT)organisatie heeft zijn invloed op alle onderliggende organisatieonderdelen en haar activiteiten, zo ook voor het testen. Dit strategisch beleid kan alle vormen hebben. Zo kan het beleid voorwaarden stellen aan de interne (IT)organisatie. Maar ook aan de kwaliteitsdoelstellingen en welke mogelijkheden er zijn waarmee die verwezenlijkt moeten worden. Strategisch beleid kan gevormd worden door wensen en eisen die van binnen de organisatie komen. Maar de mogelijkheid bestaat ook dat factoren van buiten de organisatie een rol spelen. Voorbeelden hiervan zijn eisen die gesteld worden vanuit externe toezichthouders, (lokale) wettelijke verplichtingen waaraan voldaan moet worden of brancheafspraken[1].

Tactisch

Op het tactisch niveau wordt het strategisch beleid vertaald naar hoe het in de operatie uitgevoerd gaat worden. Dit gebeurt door middel van het opstellen van voorschriften. Deze voorschriften geven aan binnen welke randvoorwaarden en normen de mensen, de middelen en de methoden moeten worden ingezet om de strategische doelstellingen te realiseren. Ze beschrijven hoe binnen een organisatie, afdeling, productgroep, programma of project (afhankelijk van de inrichting van de betreffende organisatie), inhoud gegeven moet worden aan een gestructureerde testaanpak.

Operationeel

Op het niveau van de operatie kan onderscheid worden gemaakt tussen ondersteuning van het testproces en de werkelijke uitvoering van het testproces. Voorbeelden van ondersteuning zijn onderwerpen rond methodisch, technisch en functioneel advies aan de testers. Voorbeelden van de uitvoering zijn het testen zelf en testmanagement. Deze kunnen ieder op hun eigen wijze georganiseerd zijn. Zo kunnen ze in een matrixorganisatie worden uitgevoerd, maar ook in projectverband of gedeeltelijk in de lijn.

Voorbeeld 1

Binnen een ministerie bestaan verschillende organisatieonderdelen die ieder hun eigen diensten zelfstandig aanbieden en uitvoeren. Ieder organisatieonderdeel heeft zijn eigen lokale IT-afdeling ter ondersteuning van deze diensten (met ieder hun eigen specifieke aanpak voor testen). Op strategisch niveau werd het beleid gevormd om de ondersteunende IT sneller te kunnen aanpassen als gevolg van wijzigingen in diensten (door bijvoorbeeld wettelijke veranderingen). Op tactisch niveau vertaalde dit zich in de samenvoeging van alle lokale IT-afdelingen en deze te centraliseren tot één grote IT-afdeling. Binnen deze IT-afdeling is het testen belegd in een permanente organisatie die volgens vaste methoden en technieken werkt.

Voorbeeld 2

Een pakketleverancier stelt in haar strategisch kwaliteitsbeleid dat de kwaliteit van alle (software)producten die worden uitgeleverd aan klanten, gecontroleerd wordt door een onafhankelijke externe organisatie. Dit beleid heeft zijn invloed op hoe op tactisch niveau het testen wordt ingericht. Zo is er een overall testcoördinator benoemd die de “linking pin” vormt naar de externe organisatie.

Voorbeeld 3

De centrale bank controleert als externe toezichthouder de soliditeit en integriteit van financiële instellingen. Belangrijk onderdeel vormt hierbij het stellen van eisen aan de kwaliteit van de IT van een financiële instelling en de controle hierop. Een financiële instelling zal het voldoen aan deze eisen verwoorden in haar beleid. Op tactisch niveau vertaalt dit zich in eisen die gesteld worden aan het gebruik van productiedata als testgegevens.

Leeswijzer

Het testbeleid heeft op operationeel niveau de meeste invloed op een tester. In dit hoofdstuk wordt daarom op operationele niveau beschreven hoe een permanente testorganisatie ingericht kan worden. Dit staat in paragraaf 8.3 “Permanente testorganisatie”. Twee andere elementen die buiten een project ingericht en beheerd kunnen worden zijn testomgevingen en testtools. Deze worden in respectievelijk paragraaf 8.4 “Testomgevingen” en paragraaf 8.5 “Testtools” beschreven. Als laatste beschrijft paragraaf 8.6 “Testprofessionals” het HRM van testprofessionals. Hierbij komen functies en opleidingen aan bod, die binnen een testorganisatie kunnen worden geïdentificeerd.

8.3 Permanente testorganisatie

8.3.1 Inleiding

Een permanente testorganisatie is een andere manier van het organiseren van het testproces dan tot zover is besproken. In de komende paragrafen worden deze verschillen besproken. In paragraaf 8.3.2 “Permanente testorganisatie toegelicht” wordt het fenomeen van de permanente testorganisatie beschreven en gedefinieerd. In paragraaf 8.3.3 “Voordelen, voorwaarden en aandachtspunten” worden de specifieke voordelen beschreven en de aandachtspunten toegelicht die komen kijken bij deze manier van organiseren van het testproces. De diensten die een testorganisatie kan aanbieden en de manier waarop afspraken worden gemaakt over deze diensten komen in paragraaf 8.3.4 “Leveren van testdiensten” aan de beurt. Hierna (paragraaf 8.3.5 “Algemene procesmodel”) zal een algemeen procesmodel worden beschreven, waarna dit voor twee veelvoorkomende vormen van permanente testorganisaties verder wordt uitgewerkt (paragraaf 8.3.6 “Twee veelvoorkomende vormen van testorganisaties”). Deze twee veelvoorkomende vormen worden verder beschreven in paragraaf 8.3.7 “Test Expertise Centrum (TEC)” en paragraaf 8.3.8 “Testfabriek (TF)”). De toelichting op de rol van de permanente testorganisatie bij het fenomeen outsourcing (van de gehele IT of alleen het testen) gebeurt in paragraaf 8.3.9 “Rol van een permanente testorganisatie bij uitbesteding”. In de laatste paragraaf (paragraaf 8.3.10 “Opzetten van een testorganisatie”) worden de fasen en stappen besproken die uitgevoerd moeten worden wanneer gekozen wordt voor de inrichting van een permanente testorganisatie in de lijn.

8.3.2 Permanente testorganisatie toegelicht

Organisaties kunnen er voor kiezen om een permanente testorganisatie in te richten waar vanuit verschillende testdiensten aangeboden worden. In een permanente testorganisatie wordt dan, in tegenstelling tot projectmatig testen, niet per project invulling gegeven aan een bepaald element van het testproces, maar over alle projecten heen. Dit betekent niet dat er binnen de projecten helemaal geen testactiviteiten meer worden uitgevoerd.

Definitie

De permanente testorganisatie is een lijnorganisatie die testdiensten aanbiedt.

Er bestaat geen vaste lijst van activiteiten die uitgevoerd kunnen worden door de permanente testorganisatie. Dit is voor elke organisatie verschillend. Zo kan ervoor gekozen worden om de gehele voorbereiding en uitvoering van de test te verplaatsen naar een permanente testorganisatie. Maar ook het inrichten en beheren van de testomgeving of het inrichten en beheren van testtools kan worden uitgevoerd door de permanente testorganisatie.

Het gevolg is dat ieder proces rond dat specifiek deel van testen (testuitvoering, testomgeving, testtools enz.) wordt uitgevoerd volgens een vaste werkwijze en met herbruikbare middelen. Ongeacht welk project zal het proces dezelfde kwaliteit hebben. De permanente organisatie levert deze elementen aan projecten via zogenaamde testdiensten.

Hoe de elementen van het testproces door de permanente testorganisatie worden ingevuld (en dus welke testdiensten het

levert) hangt af van de klant. Testdiensten worden afgenomen door klanten van de permanente testorganisatie. De klantwens bepaalt welke diensten er aangeboden worden. De klant kan vele vormen hebben en dat bepaalt ook dienst die geleverd wordt. Bijvoorbeeld bij het leveren van een testomgeving kan de klant de specifieke tester zijn in een project. Maar het kan ook de ontwikkelaar van de applicatie zijn die wat wil uitproberen. Een ander voorbeeld is een dienst rond testautomatisering. Hier kunnen ook weer verschillende klanten voor bestaan. Het kan de tester zijn die een omgeving wil hebben waarin hij zijn handmatige testscripts kan automatiseren, maar ook kan het de eigenaar van een applicatie zijn die een geautomatiseerde testset voor regressie wil hebben. Kortom, klanten komen, net zoals de verschillende diensten, in alle vormen voor.

Naast de klant kan ook de doelstelling van een permanente testorganisatie de testdiensten bepalen. De doelstelling (vaak vertaald in een missie) geeft het ambitieniveau weer en is afhankelijk van vele factoren. Zo kan de organisatie een opdracht van buiten hebben meegekregen (“zorg er voor dat alle testomgevingen efficiënt wordt opgezet en beheerd” of “biedt een oplossing voor alle vraagstukken rond testautomatisering”). Maar ook kan een organisatie zelf een doelstelling creëren (“alle regressietesten worden door ons uitgevoerd”). De doelstelling van een organisatie kan in de loop van de tijd veranderen en dus het aanbod van testdiensten ook.

De permanente testorganisatie zal zo ingericht moeten worden dat het de diensten (één of meer) optimaal kan aanbieden. Inrichten gebeurt door middel van het definiëren van taken, bevoegdheden en verantwoordelijkheden in functies voor de medewerkers. Daarnaast zullen er processen worden opgesteld, ingericht en gekozen worden voor een organisatiestructuur waarin dit alles tot zijn recht komt. Hier bestaan geen vaste voorschriften voor en de ideale inrichting bestaat niet.

8.3.3 Voordelen, voorwaarden en aandachtspunten

Voordelen

Meestal is een reden om te kiezen voor een permanente testorganisatie de doorlooptijdverkorting, kostenbesparing en kwaliteitsverhoging van het testproces. Dit moet worden bereikt door bepaalde elementen van het testproces over alle projecten heen te regelen in tegenstelling tot een projectmatige aanpak. Maar wat zijn nu precies de voordelen van een permanente organisatie?

Optimale benutting van (schaarse) expertise

Testexpertise binnen een organisatie is vaak schaars. Dit geldt niet alleen voor expertise rond het opzetten en uitvoeren van testscripts maar ook bijvoorbeeld voor de inzet van testtools of het inrichten van testomgevingen. Door alle aanwezige expertise te bundelen en alle vragen voor deze expertise samen te laten komen op die ene zelfde plek krijg je een optimaliserende werking op meerdere fronten. Zo kan de gevraagde expertise veel beter gevonden worden doordat er zicht is op alle aanwezige expertise. Daarnaast kan aanwezige expertise beter verdeeld worden over de verschillende vragen. En in rustige momenten kan er gekeken worden naar de andere aanvragen. Zo kunnen bijvoorbeeld medewerkers met verstand van testtools voor meerdere projecten tegelijk aan de slag.

Voorspelbare kwaliteit van de betreffende producten

De diensten aangeboden door de testorganisatie zijn gestandaardiseerd. Hiervoor bestaan producten en processen met geldende normen. Een dienst is dus niets nieuws, het is eerder gedaan en elke medewerker weet hoe deze uit te voeren. Natuurlijk is iedere opdracht, anders maar deze variatie in de dienst is tot een minimum beperkt. Hierdoor kan de kwaliteit van een dienst voorspeld worden. Bijvoorbeeld de dienst inrichten testomgeving kan een routinematige klus worden. Dit omdat het proces met bijbehorende sjablonen, tools, technieken en checklists van de plank gehaald kan worden. En omdat het door ervaren en gekwalificeerde medewerkers wordt uitgevoerd.

Geringe opstarttijd

Doordat de diensten gestandaardiseerd zijn, is de opstarttijd tot een minimum beperkt. Er hoeven geen cursussen meer gevolgd te worden, bepaalde zaken eerst geprobeerd te worden of gezocht te worden naar de beste aanpak. De opstarttijd van een dienst als het inrichten van een testmanagement tool kan zo beperkt worden tot het minimum. Alles ligt op de plank en alles is aanwezig. Het is alleen een kwestie van het bepalen van de specifieke eisen en wensen van de klant, deze inregelen, en dan kan gestart worden.

Continue verbetering van het proces ingebed in de organisatie

De geleverde diensten zijn de verantwoordelijkheid van de testorganisatie en worden uitgevoerd door de testorganisatie. Er kan voor gekozen worden om elke uitvoering te laten eindigen met een evaluatie van de geleverde dienst. Het geleerde uit deze evaluatie stroomt terug de organisatie in en wordt verwerkt in de (nieuwe versie van de) dienst. Deze formele evaluatie en verwerking van de resultaten moet ingebed zijn in de processen van de testorganisatie.

Consolidatie en ontwikkeling van ervaringen

Het bundelen van alle aanwezige expertise in combinatie met het voorgaande waarbij ervaringen worden geëvalueerd en verwerkt zorgt voor een continu lerend effect. Dit lerende effect wordt nog eens versterkt door de kruisbestuiving tussen de verschillende medewerkers.

Kosten en doorlooptijd zijn beter planbaar

Zoals al eerder gezegd, de diensten zijn gestandaardiseerd, de medewerkers hebben de expertise en de ervaring om deze diensten uit te voeren. Het wiel hoeft niet meer uitgevonden te worden en er is duidelijk zicht op wat iets kost, wat het oplevert en hoe lang het duurt.

Kostenvoordeel door centralisatie en schaal

Wanneer er op verschillende plekken in de organisatie wordt getest zal vaak ook op verschillende plekken met externe leveranciers onderhandeld worden. Door het centraliseren van alle testers en testgerelateerde activiteiten kan de testorganisatie één centrale prijsafspraak maken met de verschillende leveranciers. Dit kunnen leveranciers zijn van tools en testers. Daarnaast zorgt het bijbehorende schaalvoordeel voor een versterking in de onderhandelingspositie en kan er dus korting worden bedongen.

Voorbeeld

Bij een logistieke dienstverlener wordt op verschillende plaatsen in de organisatie geautomatiseerd getest met behulp van testtools. Iedere afdeling kocht zelf de licenties in van de testtool, voerde zelf de onderhandelingen over de prijs met de verschillende leveranciers en regelde zelf opleidingen in de gebruikte tools. Toen er gekozen werd om het opzetten en beheren van de geautomatiseerde tests te centraliseren in de organisatie ontstond er één partij die testtools gebruikte. Dit was niet alleen voordelig voor de logistieke dienstverlener zelf maar ook voor de verschillende leveranciers. Want in plaats van vele verschillende (onduidelijke) aanspreekpunten hadden zij ook nog maar één aanspreekpunt. Dit vertaalde zich in een bijkomende efficiëntie waarbij er minder verschillende tools gebruikt werden (van 23 naar 6). Ook de licentie- en opleidingskosten werden met 35% omlaag gebracht.

Voorwaarden

Niet in iedere (IT)organisatie kan een permanente testorganisatie worden opgezet. Er zijn een aantal voorwaarden waaraan moet zijn voldaan. Deze voorwaarden zijn:

- Om de voordelen die hiervoor staan beschreven te behalen moet een minimale hoeveelheid aan werk bestaan voor de permanente testorganisatie. Vaak zie je dit alleen bij de wat grotere organisaties;
- Er moet een cultuur bestaan waarbij het mogelijk is om formele werkafspraken te maken. Dit is nodig om als permanente testorganisatie concrete afspraken te maken met de klanten;
- De organisatie moet te maken hebben met herhaalbare processen en projecten. Alleen dan kan een testorganisaties standaard diensten aanbieden;
- De organisatie moet kunnen omgaan met centrale organisaties zoals een testorganisatie. Centrale organisaties kunnen als bedreigend worden ervaren (bijvoorbeeld door de grootte). Dit mag niet het geval zijn;

Uitgediept

De permanente testorganisatie als katalysator voor professionalisering

Veel organisaties willen professionaliseren, maar hebben daar niet het geld of de tijd voor. Een mogelijkheid is dan om op zoek te gaan naar een hefboom. Een hefboom waarbij professionalisering op één specifieke plek in de IT-procesketen direct effect heeft op meerdere onderdelen in die procesketen. Een permanente testorganisatie heeft de potentie om als een dergelijke hefboom te fungeren. Het zit namelijk op het punt waar de resultaten van alle eerdere processen in de ontwikkelketen uiteindelijk bij elkaar komen in een toetsbaar eindresultaat. De testorganisatie zit als het ware als een spin in het web. Daarom kan het uitstekend de afstemming tussen de partijen faciliteren en tegelijkertijd eisen stellen aan de kwaliteit van het proces en de output. Door een permanente testorganisatie in te richten kunnen verschillende partijen hiervan profiteren. Zo kan de testorganisatie in een vroeg stadium onduidelijke of missende requirements identificeren en potentiële problemen in het ontwikkel- of beheerproces signaleren. Daarnaast houdt de testorganisatie de betrokken partijen scherp op de gewenste kwaliteit en output waardoor het eindproduct, zoals het in een vroeg stadium is beschreven, ook daadwerkelijk wordt gerealiseerd. De testorganisatie is als het ware 'het geweten' van de IT-procesketen. In de praktijk betekent dit dus dat de testorganisatie bij alle stappen in deze keten nauw betrokken is en een adviserende rol heeft.

Aandachtspunten

Na het opzetten en operationeel krijgen van een permanente testorganisatie moet er continu aandacht worden besteed aan een aantal punten. Deze aandachtspunten zijn de voorwaarde voor het succesvol zijn en blijven van de organisatie.

Testdienstverlening

Van de diensten die door de permanente testorganisatie worden geleverd, moet continu bepaald worden of het nog aansluit op wat gevraagd wordt door de klant. De klant bepaalt het gewenste kwaliteitsniveau, niet de permanente testorganisatie zelf.

Testprofessionaliteit

De professionaliteit van een testorganisatie wordt gevormd door enerzijds de kennis en kunde (vakmanschap) van de testers, maar anderzijds ook door de stabiliteit van de populatie testers binnen de organisatie. Wanneer er in een testorganisatie continu mensen in- en uitstromen geeft dat onrust en ontstaat er geen solide basis voor kennisopbouw. Het is dus belangrijk om een tester tevreden te houden. Hiervoor bestaan hulpmiddelen als carrière- en beloningsstructuur en daar wordt in paragraaf 8.6 "Testprofessionals" dieper op ingegaan.

Hergebruik

Vaak is een belangrijke doelstelling die een permanente testorganisatie meekrijgt kostenbesparing. Een manier om dit te bewerkstelligen is het hergebruik van bijvoorbeeld testware, testgegevens en testinfrastructuur. Er zal continu aandacht besteed moeten worden aan het feitelijk hergebruik en hoe dit hergebruik zo optimaal en efficiënt in te richten.

Onafhankelijkheid

Als permanente testorganisatie is het van belang om, onafhankelijk van de rest van de IT-organisatie, objectief een oordeel te vellen over de kwaliteit van de aangeleverde software of hardware. Aan de andere kant maakt de testorganisatie (nog steeds) deel uit van een groter geheel met een groter belang. Hier vormt zich een belangrijke uitdaging die soms tegenstrijdigheden met zich mee kan brengen.

Voorkomen "over de muur"-effect

Het centraliseren van testers en de objectieve rol die ze gaan spelen kan voor een tweedeling zorgen binnen de IT-organisatie. Er kan een "wij versus zij" houding ontstaan waarbij het groter belang uit het oog wordt verloren. Gevolg kan zijn dat er niet meer samen wordt gewerkt maar producten "over de muur" worden gegooid richting de testorganisatie en omgekeerd.

Voorbeeld

Binnen een overheidsinstelling ontstond het volgende besef.

- Op veel verschillende plaatsen in de organisatie wordt getest (binnen de IT-afdeling bij verschillende projecten aan de klantkant binnen verschillende organisaties);
- Iedereen heeft zijn eigen testaanpak (gestructureerd en ongestructureerd);
- De werkdruk van testers varieert enorm (voor een release hoge tijdsdruk, na een release lage tijdsdruk) wat de situatie oplevert dat testers soms niets kunnen doen (zogenaamde “wachten op werk situatie”);
- Verloop onder testers is enorm en veel opgedane kennis gaat verloren;
- De overheidsinstelling loopt steeds vaker imagoschade op doordat publieke software niet goed werkt.

De opdracht wordt gegeven om een permanente testorganisatie op te zetten met de volgende twee uitgangspunten:

- Gebruik van marktconforme methoden en technieken;
- Een aantrekkelijke werkgever zijn.

Het doel van de permanente testorganisatie moest verhoging van de productiviteit zijn. Minder “wachten op werk” en minder verloop van testers. Eén jaar na de oprichting van de permanente testorganisatie leverde dat al 10% besparing op. Dit werd deels gerealiseerd door testers effectiever in te zetten en deels doordat er minder kennis verloren ging (en dus minder opleiding nodig was).

Aandachtspunten voor het komende jaar zijn:

- Verder professionaliseren HRM;
- Verder ontwikkelen testdiensten.

8.3.4 Leveren van testdiensten

Een testdienst is een bepaald element van het testproces waar de organisatie voor verantwoordelijk is en die wordt aangeboden aan de klant. De verschillende diensten van een permanente testorganisatiekunnen zeer divers zijn. Bovendien kan dit dienstenpalet vergroot dan wel verkleind worden bij de introductie van nieuwe diensten dan wel het stopzetten van bestaande diensten. Dit alles hangt nauw samen met het beleid dat binnen een organisatie wordt gevoerd.

Voorbeeld

Een financiële instelling kiest er medio 2000 voor een permanente testorganisatie in te richten. De diensten die deze permanente testorganisatie levert veranderen in de loop van de komende jaren:

2000 oprichting permanente testorganisatie

#Medewerkers: 7
Diensten: uitvoerend testen.
Ontwikkeling: ontwikkeling nieuwe testwerkwijze.

2001

#Medewerkers: 15
Diensten: uitvoerend testen & advies gebruik testmethode.
Ontwikkeling: implementeren testwerkwijze in projecten.

2003

#Medewerkers: 30
Diensten: uitvoerend testen, testmanagement, advies gebruik testmethode en testtooling.
Ontwikkeling: Testen integraal onderdeel proces projecten, gebruik metrieken.

2005

#Medewerkers: 5
Diensten: voorschrijven, advies en ondersteuning
Ontwikkeling: Testen onderbracht bij verschillende productteams. Permanente testorganisatie is stafafdeling geworden.

Vaststellen van mogelijke testdiensten

Het identificeren van een dienst gebeurt door de combinatie van drie aspecten:

- het (deel)element van testen;
- de activiteit die wordt uitgevoerd;
- de verantwoordelijkheid die de organisatie heeft.

Het testproces bestaat uit veel proceselementen die ieder op hun beurt ook weer uit elementen bestaan (deelproceselementen). Het identificeren van deze verschillende elementen kan op verschillende manieren Zo kan er gekeken worden naar de fasen en deelactiviteiten binnen TMap. Een andere mogelijkheid is te kijken naar de producten die opgeleverd worden. Er bestaat geen vaste lijst met elementen. Voorbeelden van elementen zijn:

- Mastertestplan
- Testscripts;
- Tests;
- Testwarebeheertool;
- De fase testuitvoering;
- Automatiseren testscripts;
- Bevindingenbeheer;
- Inrichten testomgeving.

Op ieder element kunnen vijf activiteiten worden uitgevoerd, te weten:

- Ondersteunen (presenteren, trainen, coachen, adviseren);
- Controleren (kwaliteitscontrole, uitvoeren audit, reviewen);
- Conserveren (beheren, onderhouden);
- Uitvoeren (uitvoeren, coördineren);
- Research & development (ontwikkelen, implementeren).

Voor iedere combinatie van deelelement en activiteit kan de organisatie een verantwoordelijkheid hebben. Er zijn drie soorten verantwoordelijkheden te onderscheiden. In volgorde van groter wordende verantwoordelijkheid zijn dat *Geen, Inzetverplichting, Resultaatverplichting.*

Geen verplichting

Vanuit de permanente testorganisatie wordt op basis van de gestelde vraag (indien aanwezig en beschikbaar) de benodigde expertise geleverd. Er geldt verder geen verplichting of verantwoordelijkheid over hoe deze vraag verder wordt ingevuld.

Inzetverplichting

Vanuit de permanente testorganisatie moet op basis van de gestelde vraag de benodigde expertise geleverd worden. Deze levering dient te geschieden binnen een vooraf gesteld tijdbestek. De permanente testorganisatie is verantwoordelijk voor het garanderen van de continuïteit in de expertiselevering.

Resultaatverplichting

Vanuit de permanente testorganisatie moet op basis van de gestelde vraag een resultaat geleverd worden. Deze levering dient te geschieden binnen een vooraf gesteld tijdsbestek, tegen vooraf gestelde kosten en met een vooraf gesteld kwaliteitsniveau. De permanente testorganisatie is verantwoordelijk voor het garanderen van de continuïteit in deze resultaatslevering.

ACTIVITEITEN	**ondersteunend**	presenteren	trainen	coachen	adviseren	**controlorend**	kwaliteitscontrole	uitvoeren audit	reviewen	**conserveren**
ELEMENTEN										
Testtools										
Testwarebeheertool		X	X	X	X					

Testdatatool										
Testomgeving										
ST-omgeving										
AT-omgeving										
Gestructureerd testen										
Testplan		X	X	X	X					
Logisch testontwerp		X	X	X	X					
Fysiek testontwerp		X	X	X	X					
Testscript		X	X	X	X					
Testuitvoering		X	X	X	X					

X = Dienst zonder verplichting
XI = Dienst met inzetverplichting
XR = Dienst met resultaatverplichting

Tabel 8.1: Een voorbeeld van de dienstenmatrix

De drie voorgaande aspecten van element, activiteit en verantwoordelijkheid identificeren een dienst. Door deze drie in tabelvorm weer te geven ontstaat de dienstenmatrix. De matrix beschrijft in de linkerkolom de verschillende elementen van testen. In de bovenste rij staan de activiteiten. En op de kruispunten staat de verantwoordelijkheid die de testorganisatie heeft (zie tabel 8.1 “Een voorbeeld van de dienstenmatrix”).

Bieden van service levels

Bij diensten waar bepaalde verplichtingen een rol spelen (*Inzetverplichting* en *Resultaatverplichting*) is het aan te raden vooraf af te spreken wat de norm is voor deze verplichting. Zo kan de mate van het voldoen aan deze norm een indicatie zijn voor de prestatie van de permanente testorganisatie. De norm van de verplichting wordt opgesteld in overleg met de klant en vastgelegd in zogenaamde service levels[2]. Een service level geldt voor een specifieke dienst (bijvoorbeeld de dienst “uitvoeren van de fase testuitvoering”). Dit wil niet zeggen dat voor alle diensten service levels opgesteld moeten worden. Het is afhankelijk van het beleid van een permanente testorganisaties hoe service levels tot stand komen. Zo kan er voor gekozen worden om hier maatwerk van te maken. In dat geval wordt bij iedere aanvraag de service levels bepaald in nauw overleg tussen de testorganisatie en haar klant. Tijdens dit overleg wordt gekeken naar wat de klant belangrijk vindt in termen van geld, tijd of kwaliteit. Daarna wordt bepaald hoe dit gemeten moet worden en welke uitgangspunten en randvoorwaarden moeten worden voldaan. Een belangrijk uitgangspunt vormt natuurlijk de gekozen teststrategie.

Een andere manier om als testorganisatie service levels te identificeren is door het definiëren van een aantal mogelijke opties per standaard dienstverlening. Bijvoorbeeld voor de dienst “inrichten systeemtestomgeving” kan gekozen worden uit de 3 mogelijke sets service levels X, Y of Z. Ieder van deze set komt met zijn eigen prijskaartje en eisen die gesteld worden aan de klant.

Hieronder staan enkele mogelijke voorbeelden van service levels zoals die in praktijk bij verschillende projecten en diensten zijn gedefinieerd. Per service level staat aangegeven:

- op welk *aspect* het betrekking heeft (bijvoorbeeld de kwaliteit van het pro duct van de dienst of de betrouwbaarheid van de planning van de dienst of de kosten voor de uitvoering van de dienst);
- een omschrijving van het *service level* (de verplichting die wordt aangegaan uitgedrukt in meetbare eenheden);
- de *norm van het service level* (de minimale prestatie die gehaald moet worden om te voldoen aan de service level.

Aspect	Service level	Norm van de service level
Kwaliteit	Hoeveelheid en ernst van de fouten die, ten onrechte, niet tijdens de test door de organisatie zijn geconstateerd	*Per ernstcategorie is het aantal fouten dat ten onrechte in de eerste drie maanden na de test wordt geconstateerd minder dan 3% van het totaal aantal fouten per ernstcategorie*
Betrouwbaarheid	Mate waarin activiteiten conform afgesproken plan en deadlines worden uitgevoerd	*De uitloop op overeengekomen mijlpalen bedraagt ten hoogste 5% van de doorlooptijd voor die mijlpalen*
Reactiesnelheid	Snelheid waarmee nieuwe aanvragen worden opgepakt	*1. Deelproces aanmelden opdracht wordt binnen 1 werkdag afgerond 2. Deelproces intake wordt binnen 2 werkdagen na aanmelding afgerond. Bij verzoeken tot opdrachtwijziging worden binnen maximaal 2 werkdagen de consequenties kenbaar gemaakt aan de opdrachtgever*
Kennisbehoud	Inspanning die opdrachtnemer moet besteden aan inwerken van het kernteam	*Dit bedraagt niet meer dan 5% van het totaal aantal testuren*
Kostenreductie	Gemiddelde kosten van testprojecten	*Testkosten als percentage van totale projectkosten is gemiddeld over alle projecten <= 35%*
Doorlooptijd reductie	Gemiddelde doorlooptijd van testprojecten	*Doorlooptijd van testen (op kritieke pad) als percentage van totale doorlooptijd van projecten <= 20%*

Bij het afsluiten van service levels gelden de volgende uitdagingen en valkuilen:

- De definitie van de aspecten en het service level. Welke moet er gekozen worden en hoe te voorkomen dat er te eenzijdige service levels worden opgesteld. Bijvoorbeeld alleen gericht op kostenaspect en niet op andere aspecten;
- Herkenbaarheid van het gekozen service level voor de klant en zijn organisatie. Bijvoorbeeld de te hanteren testtechnieken. Dit zegt een klant meestal niet veel;
- Het onvoldoende meetbaar zijn van het gekozen service level;
- Het feit dat meestal ook andere partijen invloed hebben op de mate waarin het afgesproken service level wordt gehaald. Bijvoorbeeld snelheid van testen hangt sterk af van kwaliteit van opgeleverde software;
- Onvoldoende analyse van oorzaken van het wel/niet behalen van het service level;
- Het ontbreken van historische gegevens, waardoor het moeilijk is om harde normen vast te stellen voor het service level;
- De norm voor het service level is niet realistisch;
- Het meten van het service level is een doel op zich geworden en geen middel meer om de kwaliteit van de dienstverlening te meten.

Kenbaar maken van de testdiensten

Een succesfactor van de permanente testorganisatie is de eenduidigheid en helderheid waarmee de organisatie haar testdiensten aanbiedt. Dit is van belang voor de klant, maar ook voor de testorganisatie. Voor de klanten is het van essentieel belang te weten wat ze kan verwachten van een testorganisatie. Welke diensten worden aangeboden en wat houden deze diensten precies in? Wat houdt bijvoorbeeld de dienst “geven van opleidingen” in? Welke opleidingen worden gegeven, hoe lang duurt deze opleiding enzovoort. Deze informatie wordt vastgelegd in een zogenaamde dienstencatalogus en deze wordt beschikbaar gesteld (in een document of bijvoorbeeld via een pagina op het intranet) aan de klanten.

In de dienstencatalogus wordt van iedere dienst de volgende kenmerken vastgelegd:

Beschrijving dienst	**Korte omschrijving van de dienst.**
Relatie met andere diensten	Een eventuele relatie die een dienst heeft met een of meer andere diensten. Bijvoorbeeld de dienst “leveren van testtoolprogrammeur” hangt samen met de dienst “installeren testtools”.
Input	De verwachte input van de klant. Bijvoorbeeld bij de dienst “opstellen MTP” wordt een risicolijst gevraagd.
Voorwaarde	De voorwaarde waaraan een klant moet voldoen wil de dienst succesvol zijn. Bijvoorbeeld dat de requirements bevroren moeten zijn bij de dienst “opstellen logische testgevallen”.
Output	De verwachte output van de dienst. Bijvoorbeeld een coaching plan bij de dienst “coachen medewerker”.
Service level	Het service level van de dienst. Bijvoorbeeld bij de dienst “beschikbaar stellen tool bevindingenbeheer” kan het service level horen dat deze binnen 12 uur na de aanvraag beschikbaar is.
Kosten	De kosten die een dienst met zich meebrengt. Dit kan vastgelegd zijn in bedragen of in uren. Bij de dienst “introductiecursus acceptatietest” kan dit bijvoorbeeld zijn 20 uur (4 uur voorbereiding, 2 dagen cursus).

Voorbeeld

Een dienst van de permanente testorganisatie X binnen een productiebedrijf is “inrichten en beheer systeemtestomgeving”. Deze is als volgt beschreven in de dienstencatalogus:

Naam dienst

Inrichten en beheer systeemtestomgeving.

Beschrijving dienst

Testorganisatie X draagt er zorg voor dat de klant tijdens de testfase de beschikking heeft over een systeemtestomgeving. Bij verstoring van deze omgeving draagt Testorganisatie X zorg voor het herstel daarvan. Een aantal componenten is standaard aanwezig. Het betreft kernsystemen als CICS, Websphere en Windows XP.

Relatie met andere diensten

Voorwaarde voor deze dienst is de afname van de dienst(en):
- Testinfrastructuur.

Afname van de dienst geldt als voorwaarde voor de dienst(en):

- Ondersteuning;
- Onderhoud.

Input

Verstoring van de beschikbaarheid van de systeemtestomgeving dient via Loket Testorganisatie X te worden aangemeld.

Output

In de afgesproken testperiode is de systeemtestomgeving beschikbaar en vindt registratie, oplossing en terugkoppeling van verstoringen hierop plaats.

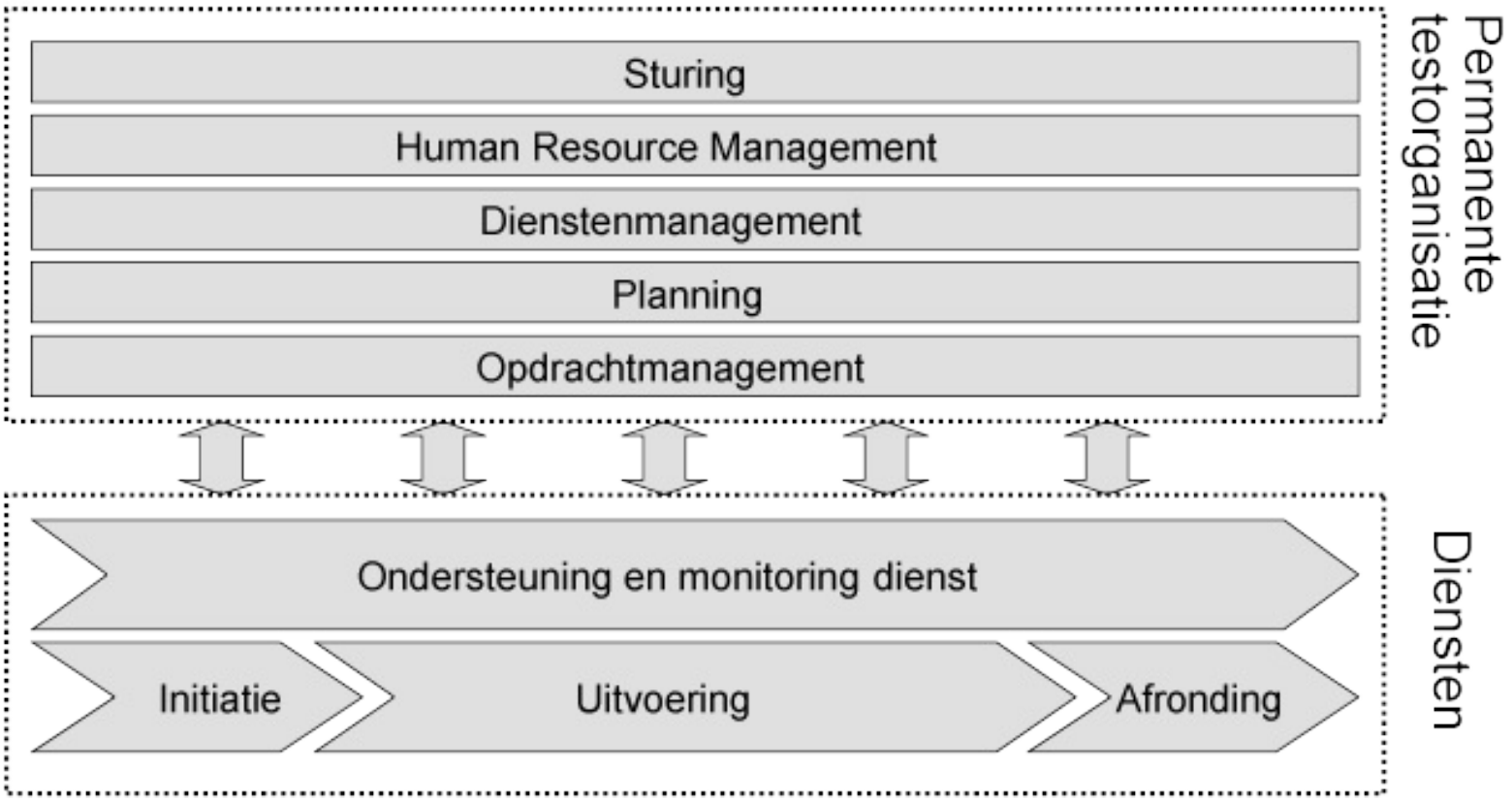
Service level

70% beschikbaarheid. Voor 100% van de bij het loket gemelde incidenten geldt dat binnen één uur na de eerste melding Testorganisatie X de status en de voortgang van het incident aan de melder terugkoppelt.

Kosten

Zie productieafspraken.

8.3.5 Algemene procesmodel



Figuur 8.1 Procesmodel van de permanente organisatie.

Voor het aanbieden van de diensten moet binnen de permanente testorganisatie een aantal processen zijn ingericht. Deze processen worden verdeelt in twee groepen. De processen voor de *diensten* dienen voor de uitvoering van de diensten die de organisatie levert. De processen van de permanente testorganisatie dienen ter ondersteuning van de uitvoering van diensten. In figuur 8.1 “Procesmodel van de permanente organisatie” wordt dit weergegeven.

Het dienstenproces bestaat uit twee parallelle hoofdprocessen: het proces voor de werkelijke uitvoering van de dienst in een opdracht en het proces dat deze uitvoering ondersteunt en monitort vanuit de organisatie. De processen zijn ter ondersteuning en richten zich op de diensten die geleverd worden, de medewerkers die de diensten leveren en de combinatie van die twee. Hieronder staan deze processen verder uitgewerkt.

Initiatie

Dit is de eerste fase voor de uitvoering van de opdracht. De opdracht bestaat altijd uit één of meer diensten van de organisatie, toegespitst op de specifieke situatie van de klant. De initiatiefase heeft tot doel het goed afbakenen van deze opdracht. Dit kan door middel van het opstellen van een zogenaamde *opdrachtomschrijving* en deze goed te laten keuren door de klant. In een opdrachtomschrijving staat concreet beschreven:

- wat de opdracht is en welke verwachtingen de klant daarbij heeft;
- de randvoorwaarden en uitgangspunten;
- de afspraken over het leveren van ondersteuning m.b.t. werkplek, methoden, opleiding en coaching vanuit de organisatie;
- de afspraken over monitoring vanuit de organisatie voor wat betreft communicatielijnen, voortgangsrapportage en -overleg;
- de op te leveren producten.

Daarnaast wordt de initiatie gebruikt om te identificeren wat er aanwezig is binnen de permanente testorganisatie om te (her)gebruiken binnen de opdracht. Denk aan sjablonen en standaarden, maar ook aan al aanwezige testscripts van vorige releases, testomgevingen of tools.

Uitvoering

Tijdens deze fase wordt de opdracht uitgevoerd volgens de afspraken met de klant. Daarnaast wordt er gecommuniceerd via de afgesproken communicatielijnen over de resultaten, voortgang, risico's en knelpunten van de uitvoering van de opdracht.

Afronding

Hergebruik van middelen is één van de succesfactoren van de permanente testorganisatie. Tijdens de afronding wordt de opdracht geëvalueerd en de tevredenheidsmeting met de klant uitgevoerd. Hetgeen geleerd wordt uit deze evaluatie en meting stroomt terug de organisatie in en wordt verwerkt in de (nieuwe versie van de) dienst. Zo ontstaat een formele procesverbetering die ingebed is in de processen van de testorganisatie.

Naast deze evaluatie wordt er aandacht besteed aan de overdracht van producten aan de testorganisatie. De producten zijn geïdentificeerd tijdens de "initiatie". Er wordt bepaald in hoeverre deze producten bewaard moeten blijven voor hergebruik. Eventueel worden de producten nog aangepast. Voorbeelden zijn sjablonen, testomgevingen, testscripts en tools.

Ondersteuning en monitoring dienst

Het opdrachtproces zoals hiervoor beschreven staat wordt continu ondersteund en gemonitord vanuit de organisatie. De voortgang, risico's en knelpunten van de uitvoering van de opdracht worden bewaakt. Daar waar nodig worden er nieuwe afspraken gemaakt over de opdracht. Ook kan er voor gekozen worden ondersteuning te leveren vanuit de organisatie aan de uitvoering van de opdracht. Dit kan in de vorm van coaching of opleiding.

Opdrachtmanagement

Binnen dit proces vinden activiteiten plaats die tot doel hebben opdrachten te verwerven voor de organisatie en de (langlopende) opdrachten te beheren. Voorbeelden van langlopende opdrachten zijn het beheer van de testomgeving of bij herhaald testen van releases. Hiervoor zal dan een contract opgesteld worden over hoe er met de opdracht wordt omgegaan door beide partijen. Hierin staan dan afspraken vermeld over het niveau van dienstverlening (de service levels) die vanuit de organisatie geleverd wordt. Andere onderwerpen zijn de manier waarop binnen de algemene opdracht (release testen) de specifieke opdrachten worden uitgevoerd (test deze release). Dit contract vervangt dus NIET de opdrachtomschrijving uit de initiatie maar kan dienen als basis hiervoor.

Planning

Het proces van planning zorgt voor de juiste medewerker op de juiste opdracht. Met “juist” wordt hier bedoeld dat aanwezige kennis en vaardigheden van de medewerker aansluiten op de benodigde kennis en vaardigheden voor de opdracht. Andere aspecten die doelen op “juist” zijn:

- de beschikbaarheid (korte termijn, is de medewerker nu beschikbaar en lange termijn, gaat de medewerker binnenkort met verlof, cursus, e.d.?);
- het carrièreperspectief (waar wil de medewerker naar toe groeien en wat voor nieuwe kennis en vaardigheden moet hij daar voor opdoen);
- de locatie (dit geldt alleen voor organisaties met meerdere geografische locaties).

Met al deze aspecten wordt duidelijk dat het planningsproces tussen alle andere processen in zit en vele belangengroepen kent. Het dient de belangen van de medewerker, het opdrachtmanagement en de gehele permanente testorganisatie.

Dienstenmanagement

Het palet aan diensten dat door de organisatie wordt geleverd, staat niet vast en kan zowel groeien als krimpen. Hiervoor moet periodiek bepaald worden of het huidige aanbod aansluit op wat gevraagd wordt en op wat de medewerkers kunnen. Daarnaast moeten diensten bekend zijn (bij klant, opdrachtmanagement en medewerker) en moeten de producten voor de diensten up-todate zijn en aansluiten op de laatste ontwikkelingen. Hiervoor kunnen verschillende activiteiten ingericht worden:

- Research (het monitoren van de markt om zo in kaart te krijgen welke nieuwe diensten ontwikkeld moeten worden en welke bestaande diensten afgebouwd kunnen worden);
- Ontwikkeling van nieuwe diensten (het ontwikkelen van nieuwe diensten met bijbehorende producten en middelen);
- Kennismanagement (het beheren van de diensten met bijbehorende producten en middelen);
- Marketing van de diensten (het bekend maken van de diensten bij de huidige en potentiële klanten).

Human Resource Management

Het proces van Human Resource Management (HRM) is er o.a. op gericht het vakmanschap (in de breedste zin van het woord) van de medewerkers van de testorganisatie voortdurend te ontwikkelen. Samen met planning ontstaat het instrumentarium voor de doorstroming van de medewerker. Hierdoor kan deze zich blijven ontplooien en binnen de beloning- en carrièrestructuur groeien. Hiervoor moeten zaken worden opgesteld als functiestructuren met bijbehorende competentie- en beloningniveaus. Daarnaast moet er aandacht worden besteed aan de individuele opleidingsplannen en evaluatie- en beoordelingsmomenten. Paragraaf 8.6 “Testprofessionals” gaat hier dieper op in.

Sturing

Sturing bevat twee aspecten; de financiële en operationele sturing. Financiële sturing is een continu proces en gebeurt op basis van budgettering (wat gaat het kosten en opleveren) en bewaking (wat kost en levert het op). Operationele sturing kan op basis van vele factoren en een voorbeeld hiervan is KPI (key performance indicator).

Een KPI is een indicator waarmee de prestatie van een organisatie of een onderdeel daarvan bepaald wordt. Er bestaat geen vaste set aan indicatoren voor testorganisaties. Het hangt voornamelijk van de doelstelling en beleid af van een testorganisatie, hoe hier invulling aan geven wordt. Enkele mogelijkheden zijn:

- het percentage medewerkers dat opdrachten voor klanten vervult;
- het percentage opdrachten dat binnen budget wordt uitgevoerd;
- het aantal defects (in relatie tot eerder gevonden defects) dat optreedt in productie;
- het percentage testdiensten dat wordt afgenomen t.o.v. testdiensten door anderen geleverd.

8.3.6 Twee veelvoorkomende vormen van testorganisaties

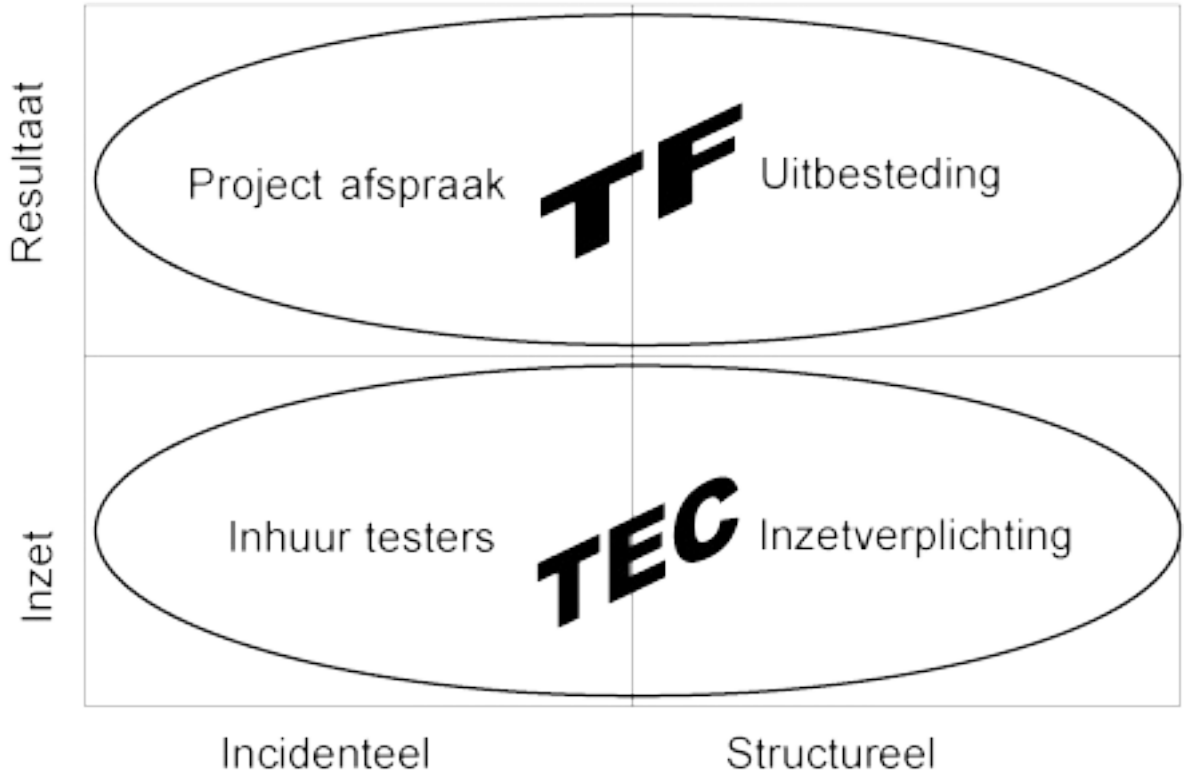
Zoals hiervoor beschreven kan een permanente testorganisatie vele vormen hebben. Zo kan er onderscheid gemaakt worden in de diensten die worden geboden. Deze keuze van diensten is weer afhankelijk van de doelstelling van de organisatie en die is op zijn beurt weer afhankelijk van vele interne en externe factoren. Er bestaat ook geen goede of foute invulling van een testorganisatie. In de praktijk zijn twee vormen van testorganisatie vaak terug te vinden. Dit zijn:

- de permanente testorganisatie als test expertise centrum (TEC);
- de permanente testorganisatie als testfabriek (TF).

Het verschil tussen deze twee zit hem onder andere in de diensten die ze aanbieden en de verantwoordelijkheid die ze daarbij hebben. Het TEC is vooral een leverende en adviserende organisatie die bij de diensten hoogstens een *inzetverplichting* aangaat. Zo kan het TEC testers of testmanagers verhuren aan een project. Of ze kunnen advies geven over een te hanteren testaanpak of een te gebruiken testtool. De activiteiten zullen altijd onder verantwoordelijkheid van het project worden uitgevoerd.

De TF echter gaat bij veel van zijn diensten een *resultaatverplichting* aan. Het proces kan worden vergeleken met een fabriek met vast personeel (testers), machines (infrastructuur), gestandaardiseerde werkprocedures, enzovoort. Verschillende klanten (afdelingen, projecten, systemen) kunnen hun complete testopdrachten uitbesteden aan deze testorganisatie. De klant komt met zijn opdracht naar de testorganisatie, de opdracht wordt ingepland in de vorm van werkopdrachten voor het personeel, de infrastructuur wordt op de correcte wijze ingesteld, de opdracht wordt uitgevoerd en de klant kan het product (rapportages, advies en mogelijke bevindingen op het geteste object) op het afgesproken tijdstip komen afhalen.

Binnen de beide testorganisaties wordt bij de testdiensten onderscheid gemaakt in de vraagfrequentie. Bij incidentele vragen (“zet een testomgeving op”) zal de testdienst anders worden ingestoken dan bij structurele vragen (“testen van releases”). Zodra het gaat om structurele vragen zullen service levels opgesteld worden. In de komende twee paragrafen wordt dieper ingegaan op beide mogelijke vormen van testorganisaties.



Figuur 8.2: Twee mogelijke vormen van testorganisaties.

8.3.7 Test Expertise Centrum (TEC)

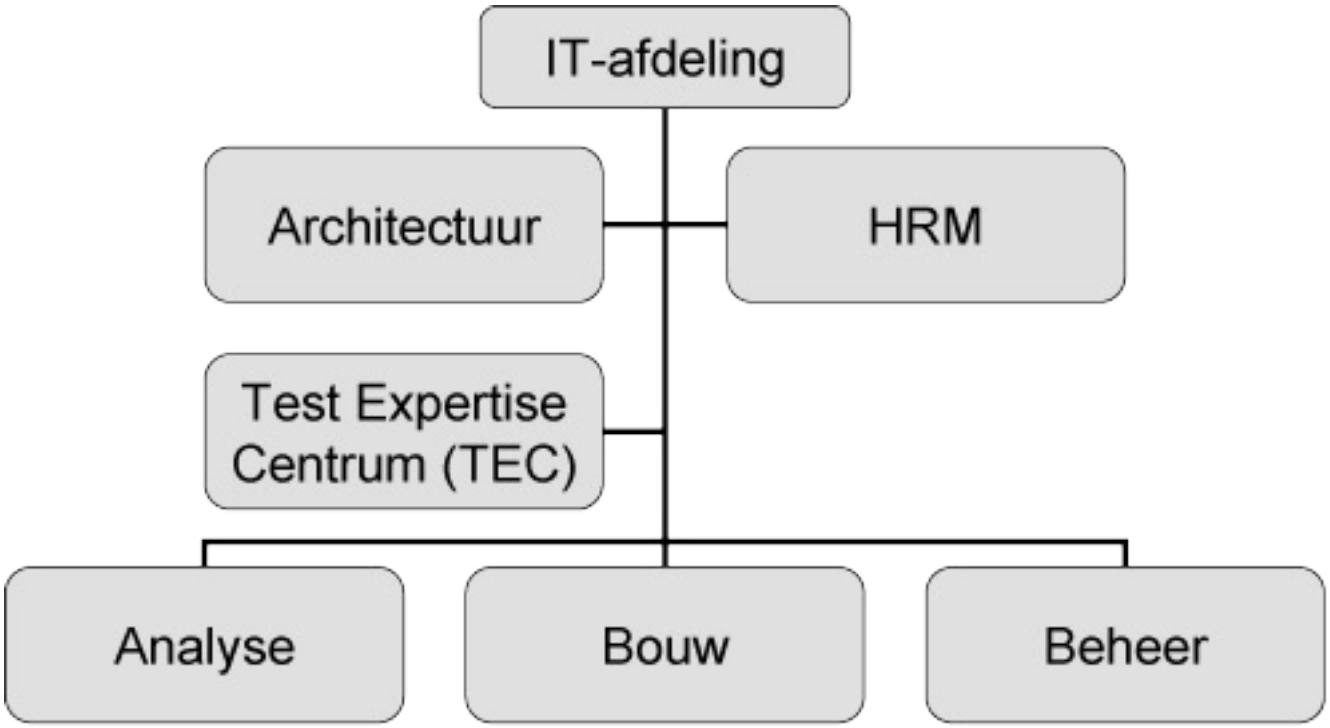
Het TEC is een permanente testorganisatie in de lijn die diensten levert aan projecten waarin getest wordt. Het vormt de interface tussen de testactiviteiten in de projecten en bijvoorbeeld organisatiebrede testvoorschriften zoals die beschreven zijn in het testbeleid. Bij het TEC ligt de uitvoering van en verantwoordelijkheid voor de activiteiten bij de projecten. Het TEC levert alleen mensen en expertise. Een belangrijke reden voor het opzetten van een TEC is efficiëntie. Zo wordt bij een goed werkend TEC voorkomen dat binnen projecten telkens nieuwe methoden, technieken en standaarden worden ontwikkeld. Het wiel hoeft dan niet steeds opnieuw uitgevonden te worden.

Naast deze levering en ondersteuning aan projecten wordt het testbeleid structureel verankerd. Zo worden de universele middelen beheerd en wordt gecontroleerd op het juiste gebruik daarvan binnen de verschillende projecten. Een ander bijkomend voordeel van een TEC is de bundeling van de meestal schaarse testexpertise. De manier waarop een TEC

wordt ingericht en georganiseerd hangt af van zaken als het testbeleid, het type organisatie waarbinnen het TEC opereert en de volwassenheid.

Organisatie

Hoe een TEC georganiseerd is kan nogal verschillen per organisatie. Zo kan een TEC alleen een groep van testadviseurs onder zijn hoede hebben en zitten de testers zelf in andere organisatieonderdelen. De testers zitten dan bijvoorbeeld in de afdelingen die per systeem zijn gevormd. Ze worden dan rechtstreeks door de projecten ingehuurd en de testadviseurs zien toe op het juiste gebruik en toepassing van de testmethoden, -technieken en -tools. In figuur 8.3 “Mogelijke organisatorisch plaats van een TEC” is het TEC als stafafdeling afgebeeld binnen de IT-afdeling.



Figuur 8.3: Mogelijke organisatorisch plaats van een TEC.

Een andere variant kan zijn dat het TEC zorg draagt voor de verhuur van testers aan projecten. Er kunnen dan testers lijnmatig onder het TEC zitten of er kunnen eventueel externe testers ingehuurd en verhuurd worden. Het aangaan van de zakelijke relaties met leveranciers van deze testcapaciteit zal dan de verantwoordelijkheid zijn van het TEC. Zo kan er voor gezorgd worden dat het testen efficiënt verloopt zonder verlies van continuïteit en vereiste kennis.

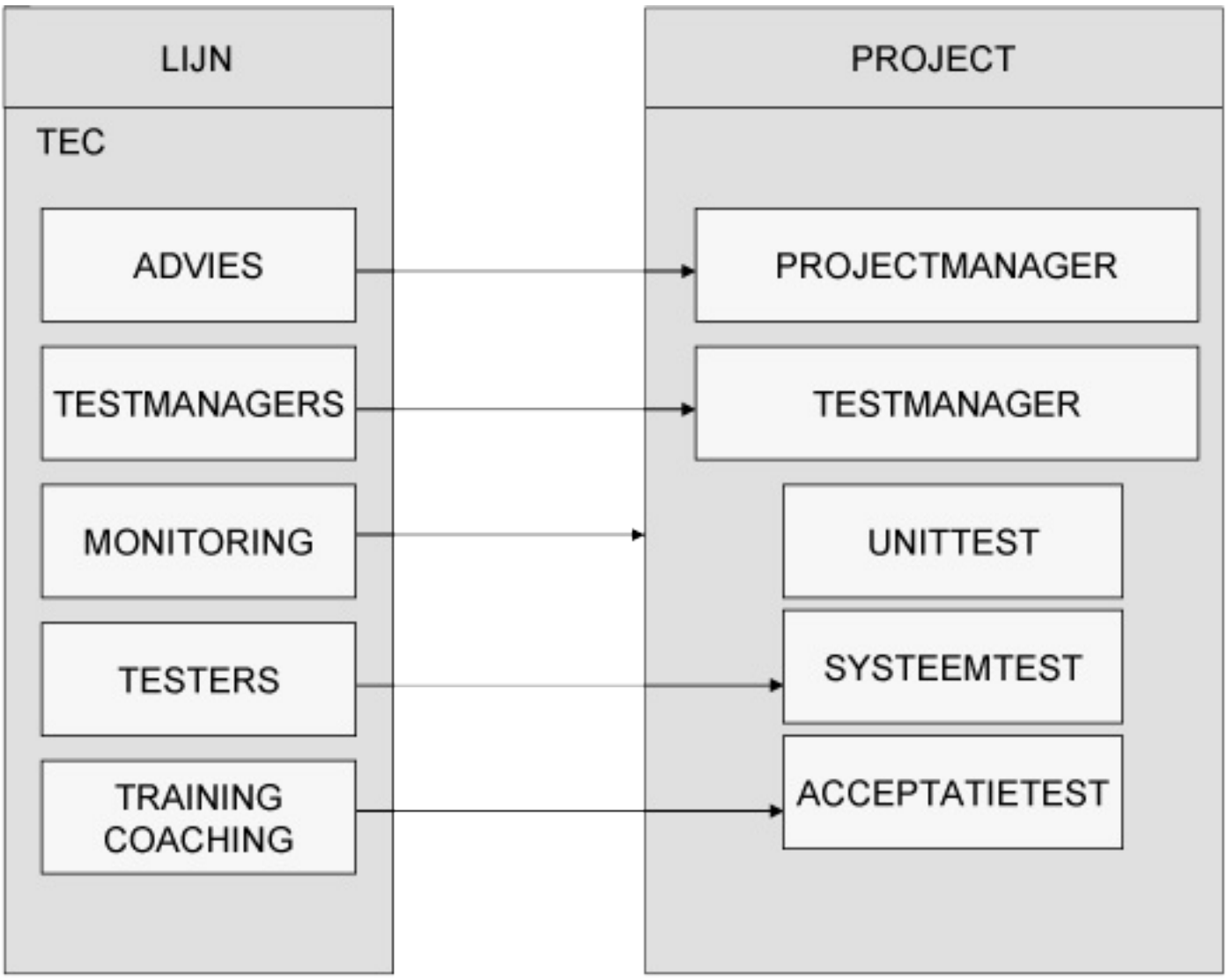
Processen

Het dienstenproces van een TEC is gericht op het leveren van medewerkers met een bepaalde expertise. Daarom zal de focus bij de initiatiefase liggen op het vinden van de juiste medewerker voor een opdracht. Hiervoor wordt in de opdrachtomschrijving vastgelegd, wat de verwachtingen van de klant zijn ten aanzien van de aanwezige expertise bij een gewenste medewerker. De evaluatie heeft tot doel de nieuw opgedane expertise (o.a. in de vorm van testware) terug de organisatie in te krijgen zodat deze op een later tijdstip weer hergebruikt kan worden. Het ondersteunen en monitoren tijdens de uitvoering van de opdracht focust vooral op het juiste gebruik van methoden en technieken. Sturing van de TEC zal voornamelijk plaatsvinden op basis van het aantal “verhuurde” medewerkers en de tevredenheid van de klant. Daarnaast moet ook goed gekeken worden naar nieuwe ontwikkelingen binnen de organisatie. Bijvoorbeeld het toekomstig gebruik van nieuwe hardware of software. Hierop op moet het TEC anticiperen door vroegtijdige om- of bijscholing van de medewerkers.

Diensten

De diensten die het TEC levert zijn dus gericht op het leveren van capaciteit en expertise. Het zijn altijd diensten zonder resultaatverplichting. De enige verplichting die er kan zijn is een inzetverplichting bij het leveren van capaciteit. Voorbeelden van diensten die geleverd zouden kunnen worden door het TEC:

- Leveren van testers, testmanagers en specialisten testautomatisering;
- Geven van cursussen rond de onderwerpen testontwerptechnieken en testmanagement;
- Het reviewen van testware op verschillende aspecten;
- Adviseren rond het gebruik van tools, inrichten van testgegevens en omgevingen;
- Coachen van gebruikers in een acceptatietraject;
- Beheren van bedrijfsbrede testmethode, technieken en voorschriften;
- Acteren van een vraagbaak rond testen.



Figuur 8.4: TEC als beheerder van de testprocessen.

In figuur 8.4 “TEC als beheerder van de testprocessen” is af te lezen hoe het TEC als de beheerder van testprocessen in een organisatie kan functioneren.

In dit voorbeeld adviseert het TEC de projectmanager over het te volgen testproces en de organisatie van de test binnen het project. De testmanager en de testers voor de systeemtest worden geleverd (verhuurd) door het TEC. Daarnaast levert het diensten waarmee het de testers voor de gebruikersacceptatietest (bijvoorbeeld toekomstige gebruikers van het systeem) binnen het project traint in de betreffende testtechnieken en zal het deze groep coachen tijdens de testuitvoering. Als laatste worden alle testactiviteiten gemonitord, waarbij er op de juiste toepassing van methoden en technieken wordt toegezien.

Dienst monitoring van het testproces

In het voorgaande voorbeeld neemt de dienst monitoring van het testproces een aparte plaats in. Voor het inrichten hiervan zijn een aantal specifieke elementen te onderkennen, die als bron van informatie kunnen dienen. Zo produceert het testproces zelf een aantal testware (zoals een testplan en bijvoorbeeld testgevallen) die een indicatie geven over hoe er getest wordt. Daarnaast kan het testproces een aantal gegevens opleveren (zoals uitgevoerde tests en aantallen en type bevindingen) die meer informatie geven over de voortgang van het testen en de kwaliteit van het te testen systeem. Op basis van deze bronnen van informatie zijn er voor monitoring twee technieken opgezet:

Checklists en audits

De checklists bestaan uit vragenlijsten over de testware die het testproces moet opleveren. De belangrijkste testware zijn het mastertestplan en de teststrategie, de detailtestplannen, de voortgangsrapportage, de testgevallen, de testscripts en de eindrapportage. De checklists richten zich op de inhoud van deze producten. Anderzijds richten deze zich op de standaard volgens welke deze producten gemaakt zijn.

De checklists kunnen op twee manieren gebruikt worden. Zo kan het TEC deze checklists beschikbaar stellen aan de projecten. Deze kunnen dan worden gebruikt om inzicht te krijgen hoe goed ze het testproces uitvoeren. Het risico is dan wel dat projecten hier subjectief mee omgaan en er belangenverstrengeling ontstaat (controle van eigen werk). Een objectievere manier is om periodieke audits te laten uitvoeren door een testadviseur van het TEC die deze checklists gebruikt.

Metrics

Metrics zijn gegevens die samen zijn verwerkt tot zinvolle informatie, vaak in een grafiek of wiskundige formule. Deze metrics kunnen ook gebruikt worden om inzicht in het testproces te krijgen. Het project zal hiervoor een aantal gegevens moeten bijhouden, zoals het aantal uren, het aantal testscripts en het aantal bevindingen per ernstcategorie. Op basis van deze gegevens kan inzicht verkregen worden in de voortgang van het testen, maar ook in de kwaliteit van het testen en het testobject. Metrics als testeffectiviteit en testefficiëntie kunnen hiervoor dienen. Ook de dekkingsgraad van het testen kan als metric worden gebruikt. Uiteindelijk is het belangrijkste doel van testen het geven van inzicht in de kwaliteit van het testobject, of eigenlijk in de ontbrekende kwaliteit. De belangrijkste metrics betreffen dan ook de gegevens over de kwaliteit van het te testen systeem. Voor meer informatie over metrics zie hoofdstuk 13 “Metrics”.

De mate waarin de technieken gebruikt worden hangt samen met het type project en het type applicatie. Met de informatie uit deze technieken krijgt het TEC een goed inzicht in de risico’s van het testproces en in de prestaties van het project. Op basis van de aldus verkregen informatie kan het TEC de organisatie en het project rapporteren, adviseren en sturen op standaardisering, kwaliteit van het testproces en daarmee kwaliteit van het product.

Tip

Monitoring bij uitbesteding

De dienst monitoring neemt een hele nuttige plaats in bij organisaties waar het testen geheel of gedeeltelijk is uitbesteed (outsourcing). Zie hiervoor paragraaf 8.3.9 “Rol van een permanente testorganisatie bij uitbesteding”.

Succesfactoren

Het succesvol zijn van het TEC hangt af van vele factoren (zie ook paragraaf 8.3.3 “Voordelen, voorwaarden en aandachtspunten”). Hier staan een aantal voorbeelden genoemd:

- De continuïteit in de levering van capaciteit en benodigde expertise kan een risico zijn. Dit kan vooral optreden wanneer het TEC niet zelf de testers als vast personeel in dienst heeft maar optreedt als intermediair tussen projecten en aanbieders van testpersoneel. Dit kan dan ondervangen worden door goede relaties op te bouwen met de aanbieders en strakke afspraken te maken over inzet van testers;
- Er moet voldoende werkvoorraad zijn in de vorm van testopdrachten om de bezetting van het TEC te waarborgen. Een manier om dit te garanderen is het principe van “gedwongen winkelnering”. Bij dit principe wordt er binnen de IT-organisatie afgedwongen dat projecten alleen maar testdiensten mogen afnemen van het TEC en van geen andere (externe) partij;
- Er moet voor elke opdracht voldoende systeem- en materiekennis aanwezig zijn. Wanneer dit onvoldoende is (bijvoorbeeld in het geval van heel erg verouderde of heel erg vernieuwende systemen) kan kennis worden onttrokken van het project of van andere afdelingen;
- Er kunnen conflicten ontstaan tussen het TEC en het projectmanagement over de mate waarin testprocessen moeten worden toegepast. De TEC processen moeten daarom organisatiebreed worden gedragen en er moeten goede communicatie- en escalatielijnen worden afgesproken in geval van meningsverschillen. De coöperatieve houding die het TEC inneemt bij meningsverschillen is mede bepalend voor het succes. Het zogenaamde ‘ivoren toren’-gedrag, waarbij het TEC de positie inneemt als bureaucratische afdeling die streng controleert op allerlei regels en

voorschriften, zal meteen worden afgestraft.

Voorbeeld

Een TEC bij een middelgrote dienstverlener rapporteert maandelijks zogenaamde behaalde resultaten. Dit gebeurt door middel van het bijhouden van een resultatendossier. Op deze manier wordt in concrete gevallen duidelijk wat de toegevoegde waarde is van het TEC. Hieronder staat een deel van dit resultatendossier:

Datum	Project/Dienst	Situatie	Resultaat
12-05-2006	XYZ	Aanvraag van 5 testers gekregen	Na review van teststrategie aanvraag kunnen reduceren naar initieel 3 testers
27-05-2006	Testtools	Prijsafspraken leverancier	Na overleg met leverancier 3 licenties kunnen afstoten en mantelprijs afgesproken voor resterende 8 licenties
04-06-2006	Testuitvoering	Tijdelijke dip in aanvragen	Eigen proces op orde maken. Testers ad hoc ingezet bij release testen ABC
10-06-2006	ABC	Testmanager bij project loopt vast	Door coaching van testconsultant situatie weer vlot getrokken
…	…	…	…

8.3.8 Testfabriek (TF)

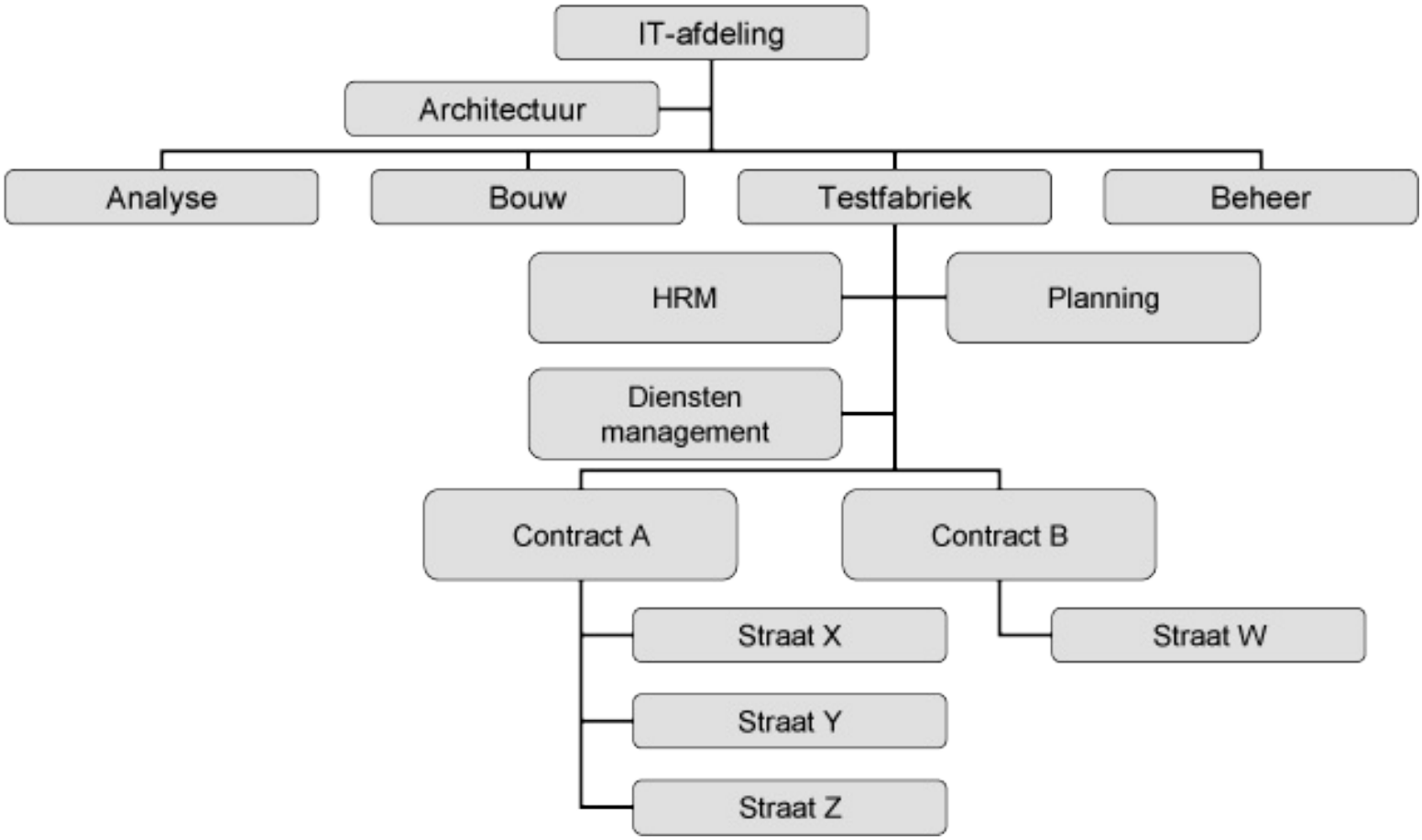
Bij een projectmatige testaanpak wordt een testtraject ingericht voor de duur van het project. Er zal dan voor het project specifiek een testproces ingericht moeten worden. De bemensing van het project dient te worden geregeld en de infrastructuur moet worden besteld en geïnstalleerd. Omdat de tester vaak maar tijdelijk aan het project is toegewezen, is het leereffect beperkt en gaat de opgedane kennis en ervaring aan het einde van het project veelal weer grotendeels verloren. Ook wordt de met zoveel moeite tot stand gekomen testomgeving aan het eind ontmanteld. Dit is geen optimale situatie waar het gaat om beheersing van kosten, tijd en kwaliteit.

Een vorm van de permanente testorganisatie, die bovengenoemde nadelen van een projectmatige aanpak opheft, is de TestFabriek (TF). Hier kunnen opdrachtgevers op structurele basis hun testopdrachten beleggen. De TF omvat een vast team van testers, werkplekken, testinfrastructuur en testtools. Binnen de TF kunnen volledige testtrajecten worden uitgevoerd (van testplan tot eindrapport), maar ook delen van testtrajecten (bijv. testspecificatie). Het testproces in een TF kan worden vergeleken met een fabriek met vast personeel (testers), machines (infrastructuur), gestandaardiseerde werkprocedures, enzovoort. Verschillende klanten (afdelingen, projecten, systemen) kunnen hun testopdrachten uitbesteden aan deze testorganisatie.

Een TF wordt georganiseerd in zogenaamde teststraten. Voor opdrachtgevers die op structurele basis testwerkzaamheden bij de TF beleggen wordt een eigen teststraat ingericht. Aan iedere teststraat is een vast kernteam van medewerkers verbonden, die zorgt voor continuïteit en kennisbehoud. Daarnaast bestaat er een flexteam. Wanneer het werkaanbod binnen hun teststraat onvoldoende is, worden zij (tijdelijk) op andere teststraten ingezet. Dit werkt dus als een flexibele pool van medewerkers die afhankelijk van het werkaanbod op een teststraat worden ingezet. In figuur 8.5 “Mogelijke organisatiestructuur van een TF” wordt een mogelijke organisatiestructuur van een TF weergegeven.

Definitie

Een teststraat is de operationele organisatie voor het leveren van testdiensten met betrekking tot één of meer opdrachtgevers. In een teststraat zit een vast team testers, infrastructuur en gestandaardiseerde werkprocedures.



Figuur 8.5: Mogelijke organisatiestructuur van een TF.

Processen

De klant komt met zijn opdracht naar de testorganisatie waar de opdracht wordt ingepland in de vorm van werkopdrachten voor het personeel. De infrastructuur wordt op de correcte wijze ingesteld, de opdracht wordt uitgevoerd en de klant kan het product (rapportages, advies en mogelijke bevindingen op het geteste object) op het afgesproken tijdstip komen afhalen. De kwaliteitsnormen van de testorganisatie garanderen de klant een constante hoge kwaliteit van het testen. Een ander voordeel is dat de veelal schaarse expertise op het gebied van gestructureerd testen, testomgeving en tools op deze wijze optimaal wordt benut. De nadruk ligt bij een TF veel meer op de lange termijn aspecten van het testproces (efficiëntie, kwaliteit) dan op een eenmalig projectaspect als het binnen tijd en geld leveren van een kwaliteitsadvies. Er wordt veel aandacht besteed aan bijvoorbeeld de herbruikbaarheid van testware, flexibiliteit van de infrastructuur, testautomatisering en evaluatie en optimalisering van de gestandaardiseerde werkwijze. Om een kernbemanning van professionele testers te krijgen, moet personeel worden opgeleid en getraind in het testen. Om de medewerkers gemotiveerd te krijgen en te houden, richt personeelszorg testfuncties in en biedt de TF de medewerkers een carrièrepad in het testen.

Het dienstenproces van een TF is gericht op het uitvoeren van diensten onder eigen verantwoordelijkheid met eigen medewerkers. Daarom zal bij de TF de initiatiefase van het primaire proces uitgebreider zijn. Er zal meer aandacht besteed worden aan voorbereidende werkzaamheden zoals het opstellen van de begroting en de intake van de benodigde middelen van de klant. Voorbeelden hiervan zijn het object dat getest moet worden, requirements, testomgeving of al aanwezige testscripts.

Ook de afronding kent een wat meer uitgebreide vorm omdat deze een interne en externe oplevering kent. De interne oplevering is binnen de TF zelf; de teststraat levert op. Hier moet een eerste controle op de kwaliteit plaats vinden. Daarna pas zal de externe oplevering richting klant plaatsvinden wat ook weer gepaard gaat met verschillende evaluaties. Daarnaast vindt er nog een verlate evaluatie plaats (na 2 à 3 maanden) voor de beoordeling van het geleverde werk op langere termijn. Het ondersteunen en monitoren tijdens de uitvoering van de opdracht focust zich, naast het juiste gebruik van methoden en technieken, vooral ook op kosten en doorlooptijd.

Beheren van testware

Een belangrijk onderdeel van de TF is het beheren van de testware. Wanneer dit goed is ingericht kan veel voordeel

worden behaald. Dit voordeel vertaalt zich in het snel kunnen opstarten van testtrajecten en efficiënt uitvoeren van de test van een nieuwe release. Het is wel noodzakelijk dat de testware up-todate wordt gehouden. Voor het goed inrichten van dit beheer moet voldaan zijn aan een aantal randvoorwaarden. De belangrijkste is dat er een beheerder moet zijn. Iemand moet de rol van beheerder hebben voor het beheer van de testware tussen de verschillende testen in. Een tweede randvoorwaarde is dat deze rol de juiste taken, bevoegdheden en verantwoordelijkheden heeft. Een derde en laatste randvoorwaarde is dat voor dit beheer het service level vastgelegd moet worden. Bijvoorbeeld hoe snel en wanneer moet de testware worden geleverd. Daar tegenover staat natuurlijk wel de voorwaarde waaraan moet worden voldaan voor het beheer van testware. Hieronder vallen zaken als het werken met bepaalde sjablonen en het werken volgens bepaalde technieken.

Het vaststellen van het voorgaande geschiedt door de beheerder in samenspraak met het (opdracht)management van de TF. Het beheren van testware is een wezenlijk onderdeel van de verschillende diensten die worden geleverd. De verantwoordelijkheid voor de uitvoering van het beheer valt onder dienstenmanagement. De volgende activiteiten dienen ten aanzien van het beheer van testware binnen een TF ingeregeld zijn:

Fase Initiatie

Activiteit 1: Vaststellen benodigde testware
De trigger om een nieuwe testopdracht te starten is de melding vanuit opdrachtmanagement dat bijvoorbeeld een nieuwe release van een systeem aangeleverd wordt. Er wordt aangegeven welke wijzigingen (technisch en functioneel) er zijn aangebracht ten opzichte van de vorige versie. Hierdoor kan worden bepaald welke testware nodig is en wanneer deze geleverd moet worden.

Activiteit 2: Uitleveren testware
De beheerder verzamelt de aanwezige testware en levert deze fysiek vanuit het dienstenmanagement aan de testopdracht. Dit kan worden vastgelegd in een uitleverdocument of paklijst.

Fase Uitvoering

Stap 3: Aanpassen testware en uitvoeren test
Nadat de testware is uitgeleverd door de beheerder wordt deze door de betrokken testers geactualiseerd op juistheid en volledigheid. Door de wijzigingen die zijn aangebracht op het systeem kan dit leiden tot aanpassingen in de testware. Deze aanpassingen worden gedaan op de versie die in bezit is van de testers, niet op de centrale versie. Aanpassingen kunnen zowel het toevoegen als aanpassen van bijvoorbeeld testgevallen zijn. Maar ook het verwijderen van testgevallen is een optie. Dit laatste is van belang om de hoeveelheid testgevallen niet nodeloos groot te laten. Anders nemen de beheersbaarheid en het overzicht af.

Fase Afronding

Stap 4: Retourneren testware
Na afronding van de testopdracht wordt de testware bevroren en fysiek aan de beheerder in het proces van dienstenmanagement overgedragen. Dit kan worden vastgelegd in een retourdocument of paklijst.

Stap 5: Archiveren testware
Deze stap betreft het zodanig archiveren van de testware dat bij een nieuwe aanvraag exact dezelfde testware teruggehaald kunnen worden. Belangrijke voorwaarde is dat de testware niet tussentijds gewijzigd kan worden zonder dat deze door de beheerder is vrijgegeven.

Dienstencontracten

Bij structurele opdrachten die op regelmatige basis terugkeren zal er gewerkt worden met een dienstencontract. Hierin zijn alle contractuele afspraken met betrekking tot de structurele opdrachten vastgelegd. Onderwerpen zijn:

- Opdrachtgever/opdrachtnemer;
- Doelstelling van de opdrachtgever;
- Scope (welke systemen, welke testsoorten, welke testdiensten);
- Plaats van uitvoering;
- Werkwijze;

- Kwaliteitsborging;
- Organisatie;
- Faciliteiten;
- Service levels;
- Randvoorwaarden en uitgangspunten;
- Financiële afspraken;
- Looptijd en beëindiging;
- Algemene voorwaarden;
- Ondertekening.

8.3.9 Rol van een permanente testorganisatie bij uitbesteding

Het komt steeds vaker voor dat competenties als testen of ontwikkelen geheel of gedeeltelijk worden uitbesteed. Soms wordt zelfs de gehele automatiseringsafdeling afgestoten en worden alle IT-activiteiten uitbesteed omdat de organisatie zich wil richten op de kernactiviteiten. Wanneer de hele uitbestede keten aan activiteiten wordt gezien als één geheel, dan is het niet aan te raden de producten van deze activiteiten zomaar in de organisatie te gaan gebruiken. Het risico, dat bepaalde functionele en niet-functionele eigenschappen niet of ten dele gerealiseerd zijn, is groot. Een ander risico is dat wat wel gebouwd is, niet voldoet aan het gewenste kwaliteitsniveau. Uitbesteding werkt alleen, wanneer er een hechte samenwerking bestaat tussen de uitbestedende organisatie (“demand” organisatie) en de leverancier (de “supply” organisatie).

Bij deze samenwerking speelt de permanente testorganisatie een belangrijke rol. Want hoe goed de processen of tussenproducten ook zijn, de eindproducten (zoals het systeem, maar ook bijvoorbeeld de procedures) dienen altijd nog getest te worden door de uitbestedende organisatie. Alleen zo kan op basis van een gedegen advies over de kwaliteit en risico’s het geleverde systeem in productie genomen worden. Wordt het testen door de leverancier onvoldoende uitgevoerd, dan worden in een laat stadium (acceptatietest of productie) een te groot aantal bevindingen gedaan, met projectuitloop of andere schade tot gevolg.

Om zeker te zijn dat zaken niet dubbel getest vergeten worden, is het absoluut noodzakelijk om de verschillende testsoorten van de demand- en supplyorganisatie goed op elkaar af te stemmen en te bewaken. Dit kan uitgevoerd worden door een testprofessional met de rol “ overall testcoördinator” (OTC, zie paragraaf 5.2.8 “Definiëren organisatie”). De rol van OTC omvat de taken het coördineren van de verschillende tests, het voorschrijven en toezicht houden op de manier van testen en het informeren en adviseren van het projectmanagement over de testvoortgang en de kwaliteit van het geteste systeem. De mate waarin gecontroleerd en bestuurd moet worden is sterk afhankelijk van het type uitbesteding. De rol van OTC kan ingevuld worden vanuit de permanente testorganisatie. Dit kan georganiseerd worden in een dienst als monitoring, zoals staat beschreven in paragraaf 8.3.7 “Test Expertise Centrum (TEC)”.

Een hulpmiddel voor de OTC is het opstellen van een mastertestplan dat alle testsoorten omvat. Standaard geldt het mastertestplan voor één project. Uitbesteding overstijgt echter meestal een enkel project. De initiële investering begint meestal pas rendabel te worden bij opvolgende releases van het systeem, dus bij het onderhoud. Om te voorkomen dat voor elke release weer opnieuw afspraken gemaakt moeten worden over het wat en hoe van het testen, zijn de Generieke Test Afspraken (GTA, zie paragraaf 5.4 “Generieke Test Afspraken”) in het leven geroepen. GTA vertonen gelijkenis met een mastertestplan en komen daar soms zelfs voor in de plaats. GTA bevatten de algemene afspraken over bijvoorbeeld het testproces, manier van begroten, procedures, communicatie, documentatie, etc. De GTA zijn feitelijk een soort contract met daarin de service levels tussen demand- en supply-organisatie. Deze zijn weer onderdeel van het contract, met daarin ook bijvoorbeeld afspraken over tijdige levering van capaciteit, reactietijden, inwerken van nieuwe mensen, prijsstelling, enzovoort.

Wanneer alleen het testproces wordt uitbesteed zijn er hele specifieke uitdagingen in deze samenwerking. Een uitdaging is de wijze waarop de testinspanning afgerekend dient te worden. Hoe moet dit worden vastgesteld? Ook aspecten als het beschikbaar hebben van materiekennis, de verantwoordelijkheid voor de testomgevingen, de beschikbare tools voor testen (en licenties) en soms wettelijk vereiste functiescheidingen vragen de aandacht.

Tip

Overall testcoördinatie is nuttig in organisaties waar het testen geheel of gedeeltelijk is uitbesteed maar het kan ook worden toegepast in gevallen waar er sprake is van vele projecten, er op verschillende locaties wordt getest of wanneer bij leveranciers wordt getest.

8.3.10 Opzetten van een testorganisatie

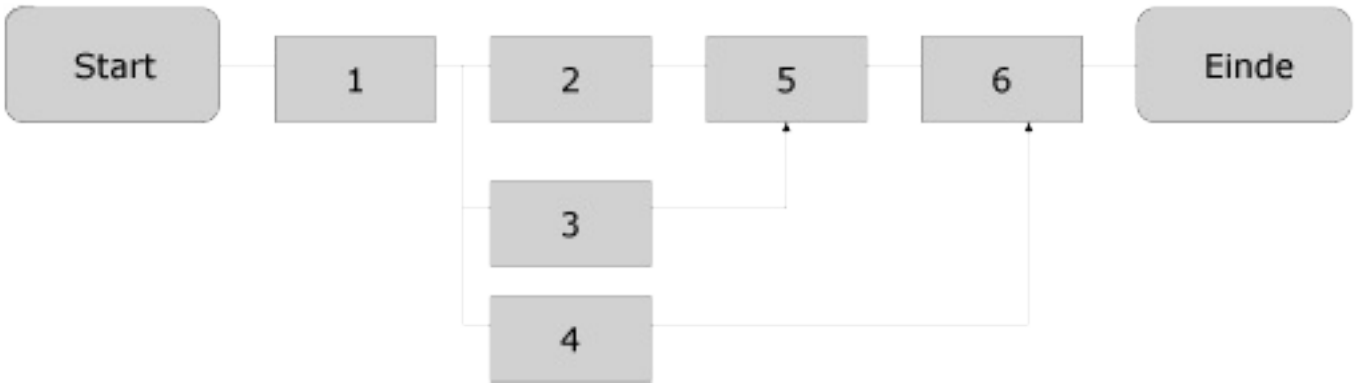
Wanneer een organisatie kiest voor een permanente testorganisatie, zal er een traject worden ingegaan waarin deze wordt opgezet. Het opzetten bestaat uit een zestal activiteiten die hierna worden besproken. Het opzetten van een permanente testorganisatie is een organisatiewijziging. Er wordt een nieuw organisatieonderdeel opgericht en in sommige gevallen zal dit als impact hebben dat mensen van organisatie onderdeel moeten wijzigen. Dit zal bepaalde consequenties hebben voor hun taken, bevoegdheden en verantwoordelijkheden. Daarom zal naast deze zes activiteiten rekening gehouden moeten worden met uitvoeren van de algemene activiteiten die bij organisatiewijzigingen horen (bijvoorbeeld afstemmen met de ondernemingsraad). Dit alles maakt dat het opzetten van een permanente testorganisatie een langdurig traject is van meerdere maanden of zelfs jaren.

Activiteiten

Onderstaand schema geeft de volgorde en afhankelijkheden aan tussen de verschillende activiteiten (figuur 8.6 “Opzetten van een testorganisatie”):

1. Inventarisatie
2. Definitie
3. Organisatie
4. Bewustwording
5. Beproeving
6. Implementatie

Het volgende schema geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten.



Figuur 8.6: Opzetten van een testorganisatie.

Werkwijze

Wanneer er interesse bestaat voor de inrichting van een permanente testorganisatie wordt er een vooronderzoek uitgevoerd. Hiertoe wordt een voorstel vooronderzoek, gemaakt waarin is aangeven hoe de probleemstelling, de oplossingsrichting en de invulling helder kan worden gekregen. Onderwerpen hierbij zijn een haalbaarheidsstudie, het organiseren van een sessie en het gebruik van de dienstenmatrix. Dit voorstel bevat een overzicht van alle betrokkenen bij deze activiteit inclusief een globale urenbegroting en planning op hoofdlijnen. Het vooronderzoek eindigt met een plan van aanpak “inrichting testorganisatie”, waarop de stuurgroep een go/nogo beslissing kan nemen voor het vervolgtraject. Aan het eind van het vooronderzoek is het plan van aanpak “inrichting organisatie” gemaakt. Dit plan bevat een globaal overzicht van de overige 5 activiteiten en een schatting van de doorlooptijden. Hierin worden de activiteiten definitie (met als belangrijk product de procesbeschrijving), organisatie, beproeving, bewustwording en implementatie toegelicht. Het plan van aanpak bevat tevens een voorstel voor het proefproject. Het proefproject eindigt met een rapport, waarop de stuurgroep een go/nogo beslissing kan nemen voor de implementatieactiviteit. Tijdens de implementatie wordt de voor een proefproject ingerichte organisatie, eventueel, bijgesteld en steeds breder in de organisatie ingezet.

Activiteit 1: Inventarisatie

In deze activiteit vindt een inventarisatie plaats van de huidige manier van testen binnen de organisatie. Het testbeleid van de organisatie waarin beschreven is hoe er wordt omgegaan met de mensen, middelen en methoden rondom het testproces in de verschillende situaties speelt hierin een belangrijke rol. De inventarisatieresultaten bevatten tevens een conclusie en een aanpak voor de volgende activiteiten; definitie, organisatie, bewustwording en beproeving. Het resultaat hiervan is vastgelegd in een plan van aanpak. De (laatste) activiteit implementatie volgt in een later stadium, na afronding van het proefproject.

Activiteit 2: Definitie

Op basis van de resultaten van activiteit 1 “Inventarisatie” worden de randvoorwaarden voor een permanente testorganisatie stap voor stap ingevuld. Een procesbeschrijving van de testorganisatie wordt opgesteld. Het resultaat is een concrete procesbeschrijving van de permanente testorganisatie voor het proefproject, inclusief de door het proefproject op te leveren producten. De door de organisatie te leveren diensten zijn eenduidig vastgelegd. De afspraken die vanuit het vooronderzoek als noodzakelijk naar voren kwamen, zijn gemaakt en bekrachtigd.

Activiteit 3: Organisatie

Op basis van de resultaten van activiteit 1 “Inventarisatie” en parallel aan activiteit 2 “Definitie” wordt de organisatie ingericht. Hoewel de hier uit te voeren activiteiten afhankelijk zijn van de resultaten van activiteit 1 “Inventarisatie”, volgt hier voor de begripvorming een mogelijke invulling. Het inrichten van de organisatie gebeurt in termen van diensten. Voorbeelden zijn testexpertise, teststraten en testwarebeheer. Voor testexpertise betekent dit het invullen van de verschillende testfuncties. In de teststraten worden díe zaken ingericht die als generiek voor een testtraject zijn te beschouwen. Het gaat hierbij om procedures, methoden, technieken, teststrategie(ën), testomgevingen, testware en testtools. De overgang van de huidige situatie naar de permanente testorganisatie is een migratieproces en wordt beschreven in een migratie(activiteiten)plan. Het resultaat is een concrete beschrijving van de organisatie en hoe deze in zijn omgeving is gepositioneerd (in- en externe organisatie). Plus een stappenplan volgens welke deze organisatie kan worden ingericht.

Activiteit 4: Bewustwording

Parallel aan een aantal andere activiteiten wordt gewerkt aan de bekendheid van de testorganisatie. Activiteiten zijn presentaties, workshops, artikelen in interne bladen, enzovoort. Belangrijk is in deze activiteit continu commitment te verkrijgen én te houden van management, klanten en uitvoerenden. De betrokkenen bij het migratieproces naar de testorganisatie op de verschillende niveaus van de organisatie moeten zich bewust zijn van de betekenis van de testorganisatie. Wanneer dit niet gebeurt, zullen de voorgestelde veranderingen nooit verankerd kunnen worden in de organisatie en zal de organisatie telkens de neiging hebben om terug te vallen in de oude werkwijze. Feitelijk hoort de bewustwording niet als aparte activiteit gezien te worden, maar eerder als essentiële preconditie. Wanneer het commitment onvoldoende is, kan beter ook niet aan het migratieproces begonnen worden.

Tip

Steun door lijnmanagement

Belangrijk in deze activiteit is dat de mensen zien dat het lijnmanagement het migratieproces steunt. Middelen om de benodigde bewustwording te verkrijgen zijn presentaties en workshops. Om die reden moet het bewustwordingsproces ook beginnen met het lijnmanagement en pas later met het projectmanagement, de testleiders en testers. Met het lijnmanagement wordt gesproken over de lange termijn doelstellingen en over de kosten/baten. Met de testers wordt gesproken over de operationele problemen en over korte termijn migratieactiviteiten. Wanneer het lijnmanagement wordt overgeslagen, loopt de testorganisatie een grote kans te mislukken, vooral wanneer de reguliere werkzaamheden en het migratieproces op een gegeven moment botsen qua prioriteiten.

Activiteit 5: Beproeving

In de testorganisatie wordt de (nieuwe) aanpak beproefd in één of meer testprojecten om zowel intern als extern de haalbaarheid aan te tonen. De resultaten van de beproevingen worden vastgelegd. Eventuele verbeteringen in zowel de organisatie als de aanpak worden doorgevoerd. Het resultaat is een ingeregelde testorganisatie voor het/de

proeftesttraject(en) met een onderbouwd go/nogo besluit voor bredere implementatie.

Activiteit 6: Implementatie

Om zo efficiënt mogelijk te kunnen werken met een testorganisatie moet deze zo breed mogelijk in de organisatie worden gebruikt. Op basis van de resultaten uit activiteit 5 “Beproeving” en ondersteund door het bereikte draagvlak vanuit activiteit 4 “Bewustwording” wordt de organisatie, (eventueel) bijgesteld, geïmplementeerd en geborgd. Nu is de permanente testorganisatie geborgd in de klantorganisatie, waar meerdere projecten voor hun testactiviteiten gebruik van kunnen maken.

Uitgediept

Valkuilen bij opzetten testorganisatie

Het succes van een permanente testorganisatie kent voorwaarden en factoren zoals die in de paragrafen 8.3.3 “Voordelen, voorwaarden en aandachtspunten” en 8.3.7 “Test Expertise Centrum (TEC)” zijn beschreven. Tijdens het opzetten van een testorganisatie zijn er valkuilen waar rekening mee gehouden moet worden:

- Tijdens het maken van de plannen is iedereen enthousiast. Wanneer mensen (projecten, managers, programmeurs, testers enzovoort) werkelijk gaan meemaken wat het inhoudt en wat voor invloed het heeft op hun dagelijkse werk, kan dit enthousiasme getemperd worden. Hier moet rekening mee gehouden worden.
- Te veel communiceren bestaat niet. Vaak wordt gezegd dat organisatiewijzigingen 20% wijzigen en 80% communiceren is. Dat geldt ook voor het opzetten van een permanente testorganisatie. Het is belangrijk om te blijven communiceren en te blijven herhalen over de testorganisatie. Wat de doelen zijn, de activiteiten, de naam, de diensten, de rol, de locatie enzovoort. Het helpt om in een vroeg stadium een naam met logo te hebben voor de nieuwe organisatie. Gebruik als communicatiemiddel alles wat voor handen is. Denk aan het intranet, het bedrijfsblad, posters aan muren en bord bij inkomsthal. Ga gedoseerd met het medium e-mail om.
- De huidige testers zijn niet beschikbaar. Een situatie die zich voor kan doen is dat de testers die voor de wijziging her en der in de organisatie zitten worden ‘weggekaapt’ door andere organisatieonderdelen (door nieuwe beloftes of nieuwe functies). Hierdoor zijn ze niet beschikbaar voor de nieuwe permanente testorganisatie met het risico dat aan de eerste verplichtingen niet voldaan kan worden. Dit wegkapen gebeurt uit angst voor het verliezen van kennis en kunde.
- Overspannen verwachtingen kunnen funest zijn voor het succes van een testorganisatie. Verwachtingsmanagement is daarom een essentieel onderdeel van het opzetten van een nieuwe testorganisatie.
- Het “wat doen al die testers hier”-syndroom. Wanneer alle testers en de testactiviteiten worden gecentraliseerd in een organisatie blijkt vaak pas hoeveel een organisatie hier aan besteedt. Gevolg kan zijn dat het management hier van schrikt en de eerste besparingsmaatregelen gaat doorvoeren. Wanneer dit gebeurt voordat de testorganisatie goed en wel is ingericht levert dit de nodige problemen op.
- Gedwongen winkelnering is een voorwaarde. Om de testorganisatie snel te laten starten is het belangrijk dat er voldoende werkvoorraad is in de vorm van testopdrachten. Een manier om dit te garanderen is het principe van ‘gedwongen winkelnering’. Bij dit principe wordt er binnen de IT-organisatie afgedwongen dat projecten alleen maar testdiensten mogen afnemen van de nieuwe testorganisatie en van geen andere (externe) partij. Het is een optie om hier een maximale periode aan te koppelen om zo de permanente testorganisatie te verplichten concurrerend te gaan werken.

8.4 Testomgevingen

8.4.1 Inleiding

Voor het dynamisch testen van een testobject (runnen van software) is een passende testomgeving nodig. Het inrichten en beheren van de testomgeving is een expertise waar testers over het algemeen geen kennis van hebben. Daarom is het inrichten en beheren van de testomgeving vaak belegd bij een aparte afdeling, buiten het project. Testers zijn wel heel erg afhankelijk van de testomgeving; zonder testomgeving is geen test mogelijk.

In deze paragraaf wordt dieper ingegaan op wat een testomgeving is en hoe de inrichting en het beheer ervan uitziet. In paragraaf 8.4.2 “Testomgevingen toegelicht” wordt gedefinieerd wat een testomgeving is, waarna in paragraaf 8.4.3

“Inrichting van testomgevingen” de eisen aan de inrichting ervan worden beschreven. Daarbij komen ook de factoren aan bod die deze inrichting bepalen. De daarop volgende paragraaf (8.4.4 “Problemen bij testomgevingen”) worden kenmerkende problemen beschreven rondom testomgevingen, gevolgd door twee oplossingen ter voorkoming van deze problemen: het OTAP-model in paragraaf 8.4.5 “OTAP-model” en drie beheerprocessen (Configuratiebeheer, Wijzigingenbeheer en Releasebeheer) in paragraaf 8.4.6 “Processen bij testomgevingen”. Hierna wordt in paragraaf 8.4.7 “Twee speciale testomgevingen” een toelichting gegeven op twee speciale vormen van testomgevingen. De aandachtspunten bij uitbesteding van de testomgeving zijn beschreven in paragraaf 8.4.8 “Testomgevingen bij uitbesteding”. Als afsluiting volgt een paragraaf (8.4.9 “Inrichten en beheren van testomgevingen als dienst”) over hoe een permanente testorganisatie een dienst rond het inrichten en beheren van een testomgeving kan opzetten.

8.4.2 Testomgevingen toegelicht

Definitie

Een testomgeving is een samenstel van onderdelen zoals hard- en software, koppelingen, omgevingsdata, beheertools en processen waarin een test wordt uitgevoerd.

Onder hardware worden alle tastbare onderdelen van een computer verstaan (scherm, harde schijf, netwerkkaart, enzovoort). Testomgevingsoftware zijn alle programma’s die op de beschikbare hardware aanwezig moeten zijn om de te testen software te kunnen runnen, zoals besturingssoftware, DBMS, netwerk en andere hulpprogramma’s. Koppelingen omvatten alles wat nodig is om het testobject met andere systemen te laten communiceren. De omgevingsdata is de set aan gegevens die de testomgeving nodig heeft om ermee te kunnen werken (gebruikersprofielen, netwerkadressen, enzovoort). Beheertools zijn tools die specifiek nodig zijn om de testomgeving operationeel te houden. Processen, tenslotte, omvatten alle activiteiten die uitgevoerd worden rond het inrichten en beheren van een testomgeving.

De inrichting en samenstelling van een testomgeving is afhankelijk van het doel van de test. Bepalend voor een goede testomgeving is de mate waarin kan worden vastgesteld in hoeverre het testobject aan de gestelde eisen voldoet. Iedere test kan een ander doel hebben en daarom kan iedere test een andere testomgeving gebruiken. Zo vraagt bijvoorbeeld een unittest een heel andere inrichting van de testomgeving dan een productieacceptatietest.

Soms is een testomgeving beperkt van omvang (bijvoorbeeld één PC bij het testen van een klein boekhoudpakket) en soms omvat het een grote verzameling van hard- en software, koppelingen en procedures, opgesteld op vele locaties (bijvoorbeeld voor het testen van het boekingssysteem van een luchtvaartmaatschappij). Behalve de testsoort en de testvorm spelen ook aspecten als de beheerstandaards, het type toepassing, de organisatiestructuur en, niet te verwaarlozen, de beschikbare budgetten een belangrijke rol.

Testomgevingen vormen een kritische succesfactor voor vrijwel elk automatiseringstraject. Hiervoor zijn verschillende oorzaken aan te dragen. Zo zijn in een productieomgeving de beheerprocessen sinds jaar en dag ingevuld en worden nog steeds verbeterd. Voor een testomgeving geldt dit niet. Processen zijn nog niet, of maar gedeeltelijk ingevuld en vaak is dit ook nog eens verschillend per afdeling en platform. De complexiteit neemt nog verder toe als in een testomgeving ook al nieuwe technologieën gebruikt worden die nog niet in productie zijn en waar dus ook minder ervaring mee is.

Een andere ontwikkeling van de laatste jaren is dat applicaties steeds meer verschillende typen hardware en software gebruiken. Bij de inrichting van een testomgeving voor dit soort applicaties vertaalt zich dit naar een keten van verschillende hard- en softwareconfiguraties met koppelingen daartussen. De metafoor van ‘de keten is zo sterk als de zwakste schakel’ laat zich dan voelen. Als één configuratie of koppeling binnen de keten verstek laat gaan, is de hele keten onbruikbaar en kan er niet volledig getest worden.

Daar komt nog bij dat een bevinding of knelpunt in een testomgeving niet altijd even snel wordt opgepakt door een beheerder. Productie gaat immers altijd voor. Er wordt dan voorbijgegaan aan het feit dat een vertraging in het testtraject ook vertraging van de in-productie-name tot gevolg heeft. Deze vertraging kan dezelfde gevolgen (of nog erger) hebben als productieverstorende fouten.

8.4.3 Inrichting van testomgevingen

Eisen die aan de inrichting worden gesteld

Bepalend voor een succesvolle testomgeving is de mate waarin kan worden vastgesteld in hoeverre het testobject aan de gestelde eisen voldoet. De inrichting en samenstelling van een testomgeving zijn dus afhankelijk van het doel van de test. Niettemin is een aantal generieke eisen te formuleren waaraan een testomgeving moet voldoen om een betrouwbare testuitvoering te garanderen.

Representatief

De testomgeving dient (zoveel mogelijk) de eigenschappen te hebben die nodig zijn voor de beoogde test. Dit betekent niet dat de gehele testomgeving altijd gelijk aan de productieomgeving moet zijn. Voor de functionele test van een koppeling (interface) tussen twee applicaties is bijvoorbeeld geen complete omgeving nodig die gelijk is aan de toekomstige productieomgeving.

> **Voorbeeld**
>
> Bij de ontwikkeling van een applicatie die uiteindelijk op een UNIX-platform moest gaan draaien werd als testomgeving voor de systeemtest een platform gebaseerd op Windows gebruikt. Het uitgangspunt was dat de functionaliteit niet beïnvloed zou worden door dit verschil in platform. Bij de GAT en de PAT werd wel een testomgeving gebruikt die op UNIX was gebaseerd.

Beheersbaar

Een beheersbare omgeving is noodzakelijk om het testobject steeds onder dezelfde omstandigheden te kunnen testen. Het moet te allen tijde duidelijk zijn welke versie er in een testomgeving is geïnstalleerd. Dit geldt niet alleen voor het testobject maar voor alle software (dus ook voor het besturingssysteem, het databasemanagementsysteem, de netwerkprotocollen enzovoort). Wijzigingen in de componenten van de testomgeving (hardware en software, testobject, procedures, enzovoort) mogen alleen na toestemming van de eigenaar van de omgeving (in projecten vaak het testmanagement) worden doorgevoerd.

Flexibel

Een testomgeving moet snel aangepast kunnen worden. Dit kan conflicteren met voorgaande eis. Het is afhankelijk van het doel van de test en de fase in het testproces welke van de twee eisen (beheersbaar of flexibel) de overhand krijgt. Aanpassingen kunnen bijvoorbeeld noodzakelijk zijn bij het analyseren van bevindingen of bij het implementeren van een nieuwe versie van de software. Ook kan het nodig zijn om bepaalde connecties met andere systemen te koppelen of juist te ontkoppelen. Als dit in een testomgeving gebeurt van één project waar verder niemand last van heeft dan wint aanpasbaarheid. In het geval van een gedeelde omgeving (bijvoorbeeld een ketentestomgeving) dan verdient beheersbaarheid de voorkeur. Andere voorbeelden van mogelijke aanpassingen zijn de systeemdatum en -tijd, valuta, rekeneenheden en regionale instellingen. Het kunnen aanpassen van de systeemdatum en -tijd kan noodzakelijk zijn om tijdsprongen tijdens het testen te maken. Dit wordt ook wel tijdreizen genoemd en op deze manier kan het systeem in het verleden of in de toekomst worden geplaatst. Hiermee kan bijvoorbeeld, in een halve dag, een systeemdoorloop van een jaar worden doorlopen. Het wisselen van regionale instellingen is van belang bij het testen van software die in verschillende landen gebruikt gaat worden.

Continu

In het geval van verstorende situaties in de testomgeving, moet zo veel mogelijk worden getracht het testen door te laten gaan. De gevolgen van een storing moeten daarom tot een minimum beperkt blijven. Een belangrijke risicobeperkende maatregel is het regelmatig maken van back-ups om deze, indien nodig, te kunnen restoren. Tevens kunnen deze veiliggestelde uitgangssituaties keer op keer worden gebruikt voor de test, of om een bepaalde bevinding te onderzoeken. Een andere risicobeperkende maatregel is om een uitwijkmogelijkheid te creëren voor de testomgeving. De uitwijkmogelijkheid kan bestaan uit een tweede logische omgeving naast de bestaande testomgeving. Het risico hiervan is dat wanneer er problemen optreden in de hardware deze beide omgevingen treft. Daarom kan er ook voor gekozen worden een tweede fysieke omgeving op te zetten. Voor het enigszins beperken van de kosten, kan er voor gekozen worden, deze te combineren met de uitwijkmogelijkheid voor productie.

Voorbeeld

Bij de aanpassing van een applicatie voor jaarlijkse contractverlengingen was het noodzakelijk om testen op verschillende data en tijdstippen plaats te laten vinden (tijdreizen). Het eenvoudig kunnen aanpassen van de systeemdatum was dus een eis die gesteld werd aan de testomgeving. Daarnaast was het door deze tijdreizen noodzakelijk om steeds back-ups te maken en later deze weer te terug te zetten. Deze combinatie van handelingen was niet complex maar zorgde bij de beheerders van de testomgeving wel voor een grote werkdruk. Daarom is toen besloten een menuscherm te ontwikkelen met daarop de verschillende handelingen en deze beschikbaar te stellen aan de testers. Op deze manier werden de beheerders ontlast en kregen de testers meer grip op hun omgeving.

Factoren die de inrichting bepalen

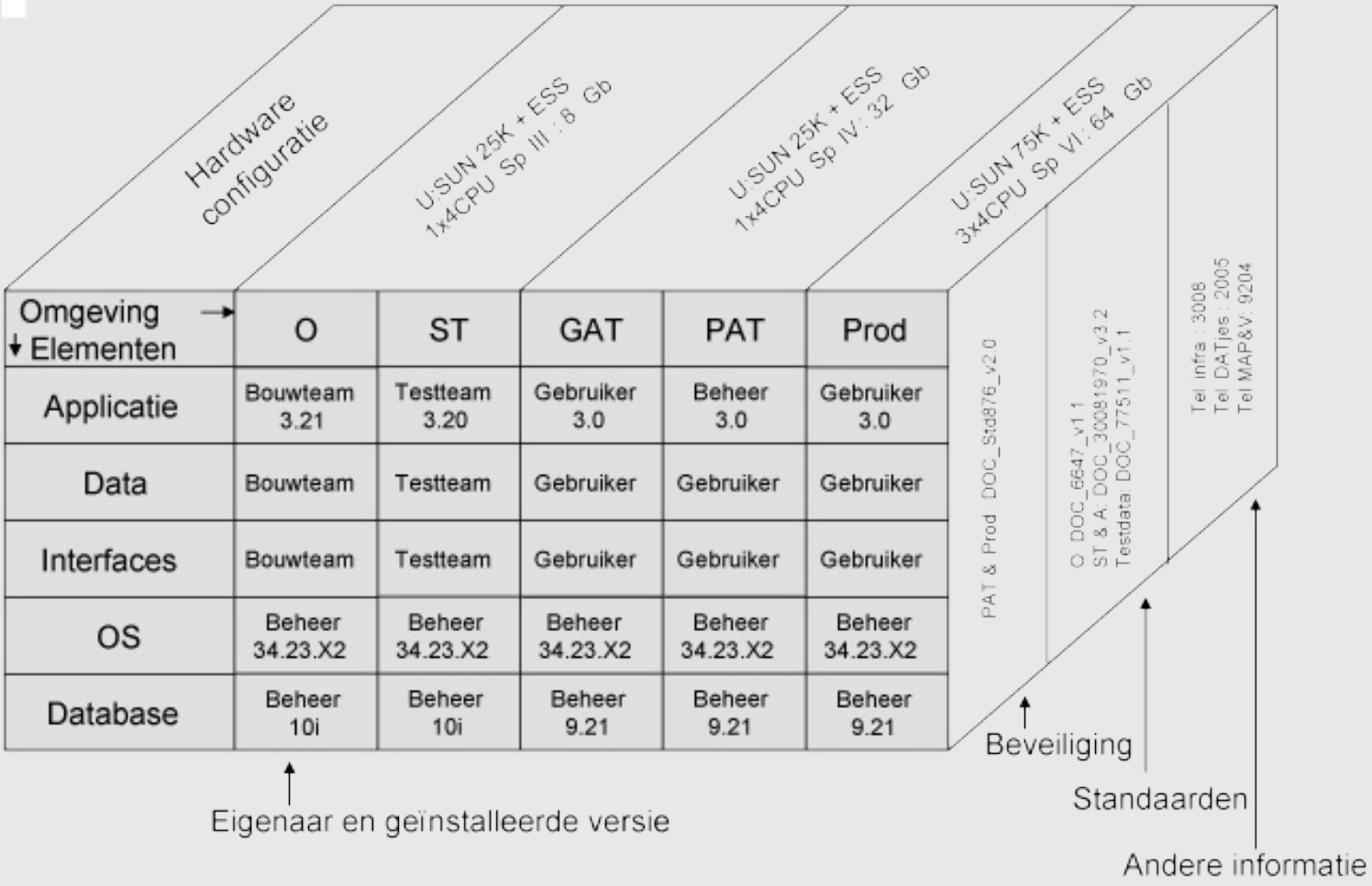
De invulling van voorgaande eisen naar de werkelijke inrichting van een testomgeving is voor elke test anders. Zo kan de testomgeving voor het testen van de schermen in de systeemtest een andere invulling hebben dan wanneer de beveiliging wordt getest tijdens de acceptatietest. Een groot aantal factoren speelt een rol bij de inrichting van de testomgeving. Hierna zijn enkele bepalende factoren opgesomd en van enige toelichting voorzien.

- De testsoort waarvoor de omgeving bedoeld is. Unit-, systeem- of acceptatietest of eventueel een gecombineerde test;
- De testvorm waarvoor de omgeving bedoeld is. Performance-, usability-, security- of regressietest?
- Eisen die aan de omgeving worden gesteld vanuit de externe organisaties. Dit kunnen toezichthouders zijn of bijvoorbeeld (lokale of centrale) overheden;
- Eisen die aan de te gebruiken testgegevens worden gesteld. Zijn dit grote of kleine volumes? Wat moet de verversingfrequentie zijn?
- Eventueel al aanwezige testomgevingen binnen de organisatie. Zijn deze bruikbaar? Hoe kunnen individuele eisen worden geëffectueerd?
- Is er een budget voor het inrichten van testomgevingen voorzien en welke mogelijkheden worden er geboden?
- Zijn er binnen de beheerorganisatie standaards voor het inrichten van testomgevingen?
- De architectuur van hardware en software. Welk ontwikkel- of productieplatform wordt gebruikt? Welke mogelijkheden worden geboden en welke belemmeringen worden daardoor eventueel opgelegd?
- De manier waarop de systeemontwikkeling is georganiseerd. De bij de systeemontwikkeling gehanteerde methoden, technieken en fasering hebben impact op de testomgevingen ten aanzien van procedures;
- Het soort systeem. De testomgeving heeft uiteraard een sterke relatie met de hoedanigheid van het testobject, bijvoorbeeld: batch, online, mainframe, PC-applicatie, maatwerk of pakket;
- De mate van gedistribueerde verwerking. In hoeverre is er sprake van datacommunicatie? In welke vorm? Maakt het netwerk of de netwerkprogrammatuur deel uit van het testobject? Worden er decentrale testlocaties gebruikt? Zijn er koppelingen met externe organisaties?
- Scope van de test. Moeten handmatige processen en bijvoorbeeld in- en output verwerking worden meegetest?
- De testomgevingen van programmeur en tester moeten geografisch niet te ver van elkaar gescheiden zijn. Hoewel communicatiemiddelen als telefoon en e-mail een deel van de communicatiebehoefte kunnen invullen, zal toch frequent overleg plaatsvinden tussen de diverse betrokkenen. Een optimale keuze van de locatie kan veel tijd en geld besparen;
- Het gebruik van testtools stelt soms eisen aan de testomgeving ten aanzien van bijvoorbeeld beveiliging, gegevensopslag en communicatiefaciliteiten.

Tip

De kubusnotatie voor testomgevingen

Van testomgevingen moeten veel kenmerken vastgelegd worden. Kenmerken die bepalend zijn voor de identificatie van een omgeving maar ook kenmerken waarover met andere partijen afspraken gemaakt moeten worden. De manier van vastleggen van deze kenmerken bepaalt mede het succes van het nakomen van de verschillende afspraken. Wanneer er meerdere testomgevingen in het spel zijn kan het duidelijk en overzichtelijk vastleggen van de kenmerken een probleem zijn. Een manier om dat te doen, is te werken met de zogenaamde kubusnotatie. Hiermee wordt in elk zichtbaar vlak van de in figuur 8.7 “Kubusnotatie van de verschillende kenmerken van een testomgeving” weergegeven kubus een aantal kenmerken gezet. Hieronder staat een voorbeeld.



Figuur 8.7: Kubusnotatie van de verschillende kenmerken van een testomgeving.

Op deze wijze is in één oogopslag alles duidelijk. Aangeraden wordt deze plaat in de gemeenschappelijke test- of projectruimte op te hangen zodat iedereen altijd inzicht heeft in de geldende afspraken.

8.4.4 Problemen bij testomgevingen

Bij automatiseringsprojecten ontstaat al snel de situatie dat er veel verschillende omgevingen gebruikt worden. Zo is vaak sprake van één of meer ontwikkelomgevingen, testomgevingen en soms ook nog beheeromgevingen.

Hiernaast zijn er nog de productieomgeving en de uitwijkomgeving. Bij deze situatie kunnen dan de volgende problemen ontstaan:

- Bevindingen die terugkomen. Een bevinding die is geconstateerd in versie X, wordt opgelost in versie X+1 en in versie X+2 treedt de bevinding plotseling weer op;
- Geen garantie dat het nog werkt. Het ontwikkelteam kan niet de garantie geven dat alles nog werkt ondanks het feit dat de release slechts uit een beperkt aantal bevindingen bestaat;
- Onaangekondigde nieuwe features. Bij het testen van een versie blijkt dat bepaalde features (nieuwe functionaliteit, bepaalde technische aspecten) al gerealiseerd zijn zonder dat de testers dat weten;
- Geen koppeling tussen bevinding en omgevingen. Een bevinding die in omgeving X is gevonden, treedt niet op in omgeving Y terwijl dit ogenschijnlijk dezelfde omgevingen zijn. Bijvoorbeeld een bevinding treedt wel op in de acceptatietestomgeving, maar niet in de systeemtestomgeving;
- Bevindingen kunnen niet onderzocht worden. Een bevinding kan niet meer onderzocht worden, omdat een andere gebruiker dan de tester, de testomgeving heeft aangepast.

Door deze problemen met omgevingen zien organisaties vaak door de bomen het bos niet meer. De oplossing voor het voorkomen hiervan is tweeledig. Ten eerste moeten de omgevingen verdeeld worden volgens het OTAP-model. Dit model en hoe het toegepast kan worden, wordt in de komende paragraaf (8.4.5 “OTAP-model”) toegelicht. Ten tweede moet rond het inrichten en beheren van de omgevingen gewerkt worden volgens formele processen. Welke processen dat zijn en hoe deze ingericht moeten worden komt in paragraaf 8.4.6 “Processen bij testomgevingen” aan bod.

8.4.5 OTAP-model

OTAP

OTAP staat voor **O**ntwikkeling, **T**est, **A**cceptatie en **P**roductie. Het model heeft als uitgangspunt dat iedere gebruiker van de infrastructuur ongestoord zijn werk wil doen en van niemand anders last wil hebben. Zo wil de eindgebruiker geen last hebben van de tester en deze wil op zijn beurt weer geen last hebben van de programmeur. Voor ieder van deze partijen is daarom een eigen type omgeving gedefinieerd. Deze 4 typen omgevingen zijn analoog aan de 4 stadia die door software wordt doorlopen: de software wordt ontwikkeld (ontwikkeling), getest (test), geaccepteerd (acceptatie) en gebruikt (productie).

Nu ziet het OTAP-model er in eerste instantie uit als een technische oplossing maar dat is het niet. Het model schrijf niet voor dat er 4 omgevingen zijn maar dat er 4 *typen* omgevingen zijn. Ieder van deze 4 typen heeft zijn eigen kenmerken. In een project (zie figuur 8.8 “Verschillende omgevingen in een ontwikkelproject volgens het OTAP-model”) kunnen dus gerust 7 omgevingen gebruikt worden volgens het OTAP-model. Zo kan het zijn dat er twee ontwikkelomgevingen zijn (lokaal en centraal), één testomgeving, twee acceptatieomgevingen (gebruikersacceptatietest- en productieacceptatietestomgeving) en twee productieomgevingen (productie en schaduw).

Omgevingstype volgens OTAP	Ontwikkeling		Test	Acceptatie		Productie	
Omgeving in ontwikkelproject	Lokale ontwikkel-omgeving	Centrale ontwikkel-omgeving	ST omgeving	GAT omgeving	PAT omgeving	Productie-omgeving	Schaduw-omgeving

Figuur 8.8: Verschillende omgevingen in een ontwikkelproject volgens het OTAP-model.

Eigenaars en beheerders van de typen omgevingen

In iedere type omgeving van het OTAP-model kunnen testactiviteiten plaatsvinden. Doordat iedere omgeving een eigenaar, beheerder en eigen groep gebruikers heeft, bezitten de verschillende testactiviteiten hun eigen karakteristieken. Zo wordt het beheer van het type omgeving test anders uitgevoerd dan het beheer van het type omgeving productie. Binnen het OTAP-model is het belangrijk om te onderkennen welke instantie nu eigenaar of beheerder is van welk type omgeving. De eigenaar is de partij die bepaalt wie de gebruikers mogen zijn en wat de beheerders moeten doen. In het OTAP-model bepaalt het doel, waarvoor de omgeving gebruikt wordt, de eigenaar. Soms is de eigenaar ook de economische eigenaar, maar dat hoeft niet altijd.

Voor het type omgeving ontwikkeling is het over het algemeen duidelijk wie eigenaar en beheerder is. Beide rollen worden ingevuld door de programmeurs. Zij schaffen de omgeving aan en beheren de omgeving. Ook voor het type omgeving productie is het duidelijk. De gebruikersorganisatie is eigenaar en vaak voert een speciale beheerorganisatie het beheer uit (in opdracht van de gebruikerorganisatie).

Bij de omgevingtypen test en acceptatie ligt het vaak wat moeilijker omdat hierbij meerdere partijen betrokken zijn. Het eigenaarschap van acceptatie ligt bij de gebruikers, het eigenaarschap van test bij de testers. Maar het beheer van de omgeving test kan op verschillende plekken liggen. Zo kan de omgeving zijn aangeschaft door, of het project, of de beheerafdeling. In dat laatste geval kan het beheer ook bij die beheerafdeling liggen, of bij de testers zelf. Het beheer kan zelfs bij de ontwikkelaars liggen.

Type omgeving	Eigenaar	Beheerder
Ontwikkeling	Ontwikkelaars	Ontwikkelaars
Test	Testers	Ontwikkelaars/ Testers/ Beheerorganisatie*
Acceptatie	Gebruikersorganisatie	Ontwikkelaars/ Testers/ Beheerorganisatie*
Productie	Gebruikersorganisatie	Beheerorganisatie

* = verschillende opties mogelijk

Tabel 8.2: De mogelijke eigenaars en beheerders van de 4 typen omgevingen.

Type omgeving	Eigenaar	Beheerder
Ontwikkeling	Ontwikkelaars	Ontwikkelaars
Test	Testers	Ontwikkelaars/ Testers/ Beheerorganisatie*
Acceptatie	Gebruikersorganisatie	Ontwikkelaars/ Testers/ Beheerorganisatie*
Productie	Gebruikersorganisatie	Beheerorganisatie

* = verschillende opties mogelijk

Tabel 8.2: De mogelijke eigenaars en beheerders van de 4 typen omgevingen.

Testvormen en de 4 typen omgevingen

Het OTAP-model dwingt geen consequente koppeling van een testvorm aan één type omgeving af. Hiermee worden nadelige gevolgen voorkomen. Door de volgtijdelijkheid van de testprocessen in de testomgevingen kunnen immers fouten te laat worden ontdekt. Dit nadelige gevolg kan worden voorkomen door een testvorm in meerdere typen omgevingen uit te voeren. Uiteraard geldt dan ook dat de oplevering van de te testen delen van het testobject gerelateerd moet zijn aan de testvorm (en de bijbehorende omgeving). In deze constructie kan het voorkomen dat de gebruiker sommige tests in de ontwikkelomgeving uitvoert.

De realisatie van dit model is een uitdaging voor het testmanagement en betrokkenen. De eigenaar van de omgeving moet er voor open staan dat zijn omgeving facilitair is voor welke testvorm dan ook. Verschillende groepen gebruikers kunnen gebruik maken van de omgeving. De hierdoor te bereiken tijdwinst door parallelliteit van de tests en reductie van de herstelkosten door vroegere detectie van fouten zijn deze inspanning meer dan waard. Het is dus vooral van belang de testomgeving passend te maken voor de testvorm en binnen het model van OTAP past dat uitstekend.

Testen in de type omgeving ontwikkeling

De unittest wordt uitgevoerd in dezelfde type omgeving als waarin de software en andere systeemcomponenten worden ontwikkeld; de omgeving ontwikkeling. De inrichting van deze omgeving en de bijbehorende testactiviteiten worden als onderdeel van het ontwikkelproces uitgevoerd. Wanneer het nodig is om een deel van de omgeving te gebruiken voor een test, is het in de meeste gevallen de ontwikkelaar zelf die hier voor zorgt. Vaak biedt het ontwikkelplatform standaardfaciliteiten voor het testen, zoals bestanden, testtools en procedures voor bijvoorbeeld versiebeheer, overdracht, bevindingen en foutherstel. Die faciliteiten bieden de ontwikkelaars dan voldoende mogelijkheden hun testproces goed te beheren. Als er geen bijzondere eisen aan de unittests worden gesteld en de genoemde standaardfaciliteiten voorhanden zijn, kunnen de tests naar behoren worden uitgevoerd. Een belangrijk aspect waarmee programmeurs te maken hebben, is de beheersbaarheid van hun omgeving. In de praktijk komt het maar al te vaak voor dat een programmeur vijf of meer versies van zijn programma onderhanden heeft. Het bewaren van de relatie tussen de testgevallen, de testresultaten en het testobject vraagt dan veel aandacht.

Testen in de type omgeving test

De omgevingstype test is er om (delen van) het gehele systeem op zowel technische als functionele aspecten te testen. Deze test moet worden uitgevoerd in een beheersbare omgeving. Beheersbaar houdt in dat er middelen voorhanden zijn waarmee onder andere de software, de documentatie, de testbestanden en de testware overgedragen en beheerd kunnen worden. Het overzetten van nieuwe of gewijzigde software moet voor de tester controleerbaar zijn. De tests moeten reproduceerbaar zijn. De individuele tests van het ene (deel)systeem moeten gescheiden van de tests van andere (deel)systemen kunnen plaatsvinden. Vooral het gelijktijdig gebruik van dezelfde testgegevens kan in dit kader voor veel problemen zorgen (zie paragraaf 6.6.2 “Definiëren centrale uitgangssituatie(s)”). In dit type omgeving kunnen tools gebruikt worden die op een technisch niveau inzicht kunnen geven over verschillende gebeurtenissen aan de tester. Voorbeelden hiervan zijn het gebruik van SQL om rechtstreeks in de database te kijken, het hebben van rechtstreekse toegang tot de log-files van het systeem en het zelf kunnen starten en stoppen van batches (zie paragraaf 8.5.3 “Soorten testtools”).

Testen in de type omgeving acceptatie

De omgeving van de type acceptatie biedt de toekomstige gebruikers en beheerders de mogelijkheid om het testobject in een omgeving te testen die zo goed als mogelijk op de productieomgeving lijkt. Doorgaans wordt de test, in dit type omgeving, gesplitst in een gebruikersacceptatietest en een productieacceptatietest. Tijdens de GAT wordt gecontroleerd of het testobject de vereiste functionaliteit levert in samenhang met productiefaciliteiten en -procedures. Tijdens de PAT

wordt gecontroleerd of het systeem voldoet aan de normen van beheer en productie, zowel ten aanzien van procedures als ten aanzien van aspecten als volumeverwerking en performance. Voor de testsoorten GAT en PAT is het aan te raden een eigen omgeving te creëren. Al is het natuurlijk mogelijk deze in dezelfde omgeving uit te voeren.

Uitgediept

De PAT-omgeving als productieomgeving

Een testomgeving voor de PAT wordt vaak duur bevonden door organisaties. Dit is ook niet verwonderlijk, want juist bij de PAT is het belangrijk dat de testomgeving niet alleen functioneel, maar vooral ook technisch hetzelfde is als de productieomgeving. Dit betekent dus logischerwijs dat een PAT-omgeving dezelfde hardware moet hebben als in productie (in soorten en in aantallen). Een PAT-omgeving is dus eigenlijk een tweede productieomgeving.

Een oplossing hiervoor kan zijn om bij nieuwbouwtrajecten de PAT-omgeving bij oplevering van het systeem te promoveren tot de productieomgeving. Zo is er maar één productieomgeving nodig. Bij onderhoudstrajecten kan ervoor gekozen worden om de PAT uit te voeren in een uitwijkomgeving die vaak een kopie is van de productieomgeving. Wanneer er geen uitwijkomgeving is, kan er ook voor gekozen worden om de PAT op de productieomgeving uit te voeren in een periode wanneer er geen gebruikers zijn (bijvoorbeeld ’s nachts of in weekenden). Deze laatste optie brengt natuurlijk wel risico’s op het vlak van beschikbaarheid van de productiesystemen met zich mee en deze is dan ook alleen aan te raden bij relatief eenvoudige systemen.

Testen in de type omgeving productie

Het testen in een omgeving die wordt gebruikt voor productie is niet wenselijk en soms zelfs verboden door toezichthouders en wettelijke instanties. In zeer uitzonderlijke situaties is het soms onontkoombaar dat in het type omgeving productie wordt getest. Dit zijn situaties waarbij de benodigde testomgeving zo complex is dat deze niet meer te simuleren of na te bouwen is. Voorbeelden hiervan zijn complexe ketens van programma’s (vaak over verschillende organisaties of soms zelfs landen heen). In dit soort gevallen kan er voor gekozen worden in productie te testen. Hiervoor moet wel het nodige geregeld worden. Zo mag de nieuwe versie van de software alleen toegankelijk zijn voor het testteam. Ook mag de uitvoering van de test het reguliere productieproces niet verstoren. Bij het testen in productie moet speciale aandacht uitgaan naar de gegevens die gebruikt worden voor de test. Zo kan gekozen worden om wijzigingen in de gegevens ongedaan te maken. Ook kunnen speciale testgegevens aan de productiegegevens worden toegevoegd, kan afgesproken worden enkel gegevens te raadplegen of kunnen de gegevens van medewerkers van de organisatie dienen als testgegevens. De uitvoering van de test zal vaak onder controle staan van een (externe) toezichthouder, omdat er handelingen worden uitgevoerd (bestellingen gedaan, betalingen uitgevoerd enzovoort) die niet formeel zijn.

8.4.6 Processen bij testomgevingen

Om de problemen die in paragraaf 8.4.4 “Problemen bij testomgevingen” staan beschreven te voorkomen, moeten processen worden ingericht voor het inrichten en beheren van de omgevingen. Deze processen zijn:

- Configuratiebeheer;
- Wijzigingenbeheer;
- Releasebeheer.

Configuratiebeheer

Het doel van configuratiebeheer is om altijd duidelijk te hebben wat de configuratie is van de verschillende omgevingen. Onder configuratie wordt naast het testobject zelf, de set van componenten bedoeld zoals ze in de definitie van testomgeving zijn opgesomd (hard- en software, koppelingen, omgevingsdata en beheertools). Hierbij gaat het niet alleen om de componenten zelf, maar vooral ook om de versie daarvan. In de CMDB (Configuratie Management Database) zijn de gegevens van deze componenten vastgelegd. Dit proces zorgt ervoor dat alleen geautoriseerde wijzigingen in de omgevingen worden doorgevoerd. Met behulp van de CMDB is eenduidig vast te stellen wat de verschillen zijn tussen de omgevingen.

Het is voor de tester belangrijk om bij dit proces betrokken te zijn om de volgende redenen:

- Om te weten in welke configuratie de bevindingen optreden. Deze configuratie moet met de bevinding zelf gerapporteerd worden;
- Om te weten hoe de versies van de testware opgebouwd moeten worden.
  Er bestaat een nauwe relatie tussen de testware en de versie van de configuraties. Het is van belang om de manier van configuratieopbouw en de manier van naamgeving en nummering over te nemen voor het beheer van de testware.

Wijzigingenbeheer

Het doel van wijzigingenbeheer is om alle individuele wijzigingen, waarover de stakeholders het eens zijn, gecontroleerd door te voeren in de verschillende omgevingen. Het doorvoeren mag niet leiden tot nieuwe bevindingen. Wijzigingen kunnen bijvoorbeeld aangeleverd worden als gevolg van testbevindingen, aanpassingen van hardware, implementatie van nieuw ontwikkelde software of aanpassing aan gegevensbestanden. Het proces van wijzigingenbeheer beziet de totale impact van een wijziging. Een actuele CMDB (zie configuratiebeheer) is noodzakelijk om de impact te bepalen in een specifieke omgeving. Om de volgende redenen is het voor de tester van belang bij dit proces betrokken te zijn:

- Wijzigingen in wensen (requirements), brengen wijzigingen in de testbasis met zich mee. Het is voor de tester dus van belang om te weten welke delen van het testobject wijzigen en wat de wijziging is, zodat eventueel de testgevallen en testscripts aangepast kunnen worden;
- Van wijzigingen wordt bepaald of ze getest moeten worden. Het kunnen testen (de testbaarheid) van een wijziging hangt onder andere af van de kwaliteit van de beschrijving van de wijziging. Wanneer een tester vroeg wordt betrokken bij de beschrijving van de wijziging kan daar op gelet worden;
- Wijzigingen leveren testwerk op en moeten dus ingepland worden.

Releasebeheer

Het doel van releasebeheer is het doorvoeren van één of meer samengevoegde en (via wijzigingenbeheer) goedgekeurde wijzigingen, van één omgeving naar de andere. Releasebeheer omvat alle componenten die onderdeel uitmaken van een wijziging.

Belangrijke onderdeel van releasebeheer is versiebeheer (de juiste versie in de juiste omgeving). Om de volgende redenen is het voor de tester van belang bij het proces van releasebeheer betrokken te zijn:

- Om te weten welke versie van de testscripts nodig zijn. Als tester is het belangrijk te weten welke versie van de verschillende componenten geïnstalleerd is. In de praktijk is dit voor de versie van de software (is meestal het testobject) het belangrijkst. Afhankelijk van de versie zijn bepaalde requirements wel of niet gebouwd en zijn dus bepaalde testscripts wel of niet nodig;
- Om te weten welke release naar de klant wordt gestuurd. De test vindt plaats op een bepaalde release en dus is het vrijgaveadvies hieraan gekoppeld.

8.4.7 Twee speciale testomgevingen

Twee varianten van de testomgeving verdienen extra aandacht: de ketentestomgeving en de productiefixtestomgeving. Dit omdat hier een aantal aspecten van belang zijn die minder spelen bij de andere omgevingen.

Ketentestomgeving

De ketentestomgeving is een omgeving waarin één of meerdere procesketens getest kunnen worden. Procesketens zijn bedrijfsprocessen die over meerdere applicaties (en vaak ook verschillende hard- en softwareconfiguraties) lopen. Hierbij geldt dat de output van de ene applicatie de input is van de andere applicatie. De tester is niet geïnteresseerd in de afzonderlijke applicaties (en hard- en softwareconfiguraties), maar ziet de gehele keten als één aaneengesloten geheel. Ketentesten vinden meestal tijdens de acceptatietest plaats.

De ketentestomgeving heeft dus een extra dimensie in de organisatorische uitdaging, die testers toch al hebben. Doordat de keten over meerdere applicaties loopt, zijn er meerdere afdelingen bij betrokken. Bovendien moeten alle componenten van de testomgeving op hetzelfde moment gereed staan en moeten er afspraken gemaakt worden rond de testgegevens in deze verschillende componenten.

Om dit goed te organiseren moeten al vooraf in het (master)testplan hier afspraken over worden gemaakt met de verschillende partijen. Het is aan te raden om binnen de testorganisatie de rol van ketenregisseur in te vullen. Deze is verantwoordelijk voor opzet en beheer van de ketentestomgeving. Tijdens de uitvoering van de ketentest zal de ketenregisseur er voor zorgen dat alle delen van ketens beschikbaar zijn en vormt hij het primaire aanspreekpunt voor de verschillende partijen. De taken, de vereiste kennis en vaardigheden van de ketenregisseur zijn gelijk aan die van de testinfrastructuurcoördinator (zie hoofdstuk 16 “Testrollen”).

Wanneer een ketentestomgeving door meerdere projecten wordt gebruikt moeten er afspraken worden gemaakt over het gebruik van testgegevens. Wanneer bijvoorbeeld project A testen uitvoert waarbij ze rekeninggegevens uit “het rekeningen bronsysteem” raadplegen, en project B heeft als opdracht “het rekeningen bronsysteem” aan te passen. Dan kan de situatie ontstaan dat project A ’s middags via een andere versie van “het rekeningen bronsysteem” gegevens ophaalt dan ’s morgens. De betrouwbaarheid van de testen van project A heeft hier uiteraard onder te lijden.

Productiefixtestomgeving

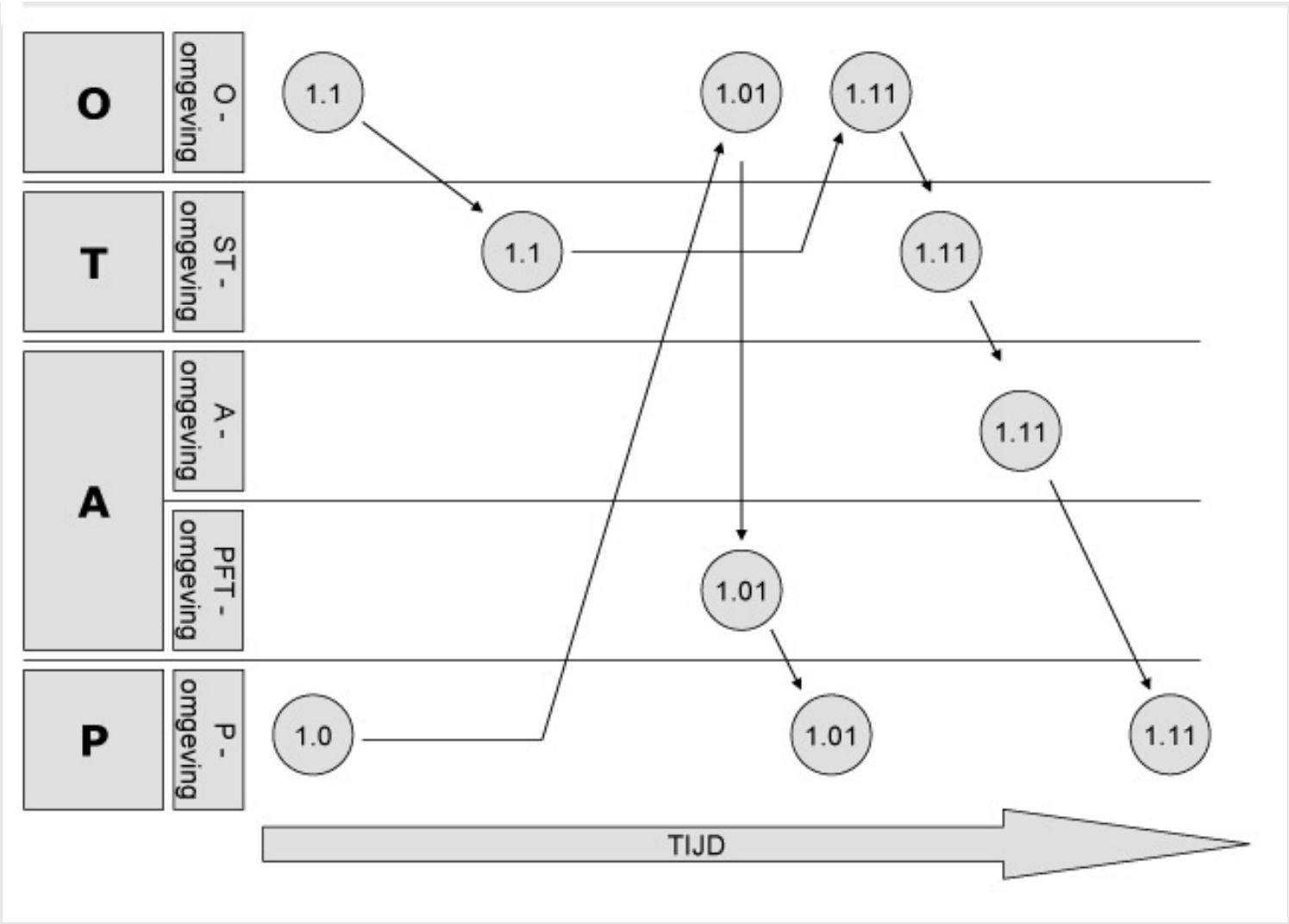
De productiefixtestomgeving is de omgeving waarin oplossingen voor productieverstorende fouten (vaak ook “fixes” genoemd) worden getest. Fixes hebben als kenmerk dat ze met de hoogste prioriteit moeten worden opgelost en getest (productieverstorend is verlies). Daarom is er geen tijd om deze ook te testen door de lijn van systeemtest en acceptatietest. Bovendien worden deze omgevingen (als ze überhaupt al per direct beschikbaar zijn) vaak gebruikt voor de test van een andere, toekomstige versie van het systeem. Een oplossing kan zijn om de fixes rechtstreeks vanuit de ontwikkelomgeving (waar ze zijn gemaakt) over te zetten naar productie, maar dit brengt grote risico’s met zich mee. Bovendien is het in sommige marktsectoren niet toegestaan waar toezichthouders en andere wettelijke instanties dit verbieden.

Een oplossing hiervoor is de productiefixtestomgeving. Deze omgeving is altijd beschikbaar of binnen zeer korte tijd beschikbaar te krijgen. Deze productiefixtestomgeving is van type omgeving acceptatie en lijkt in grote mate op de productieomgeving. Het bevat dezelfde versies van het operating-systeem, databases, netwerkprotocollen enzovoort. Ook zijn de autorisaties hetzelfde ingericht en wordt er gewerkt met testgegevens die een afspiegeling zijn van productie.

Een aandachtspunt hierbij is de beschikbaarheid van de betrokkenen van deze testomgeving. Zoals al eerder gezegd moeten productieverstorende fouten snel opgelost worden en dus kunnen beheerders en programmeurs snel opgeroepen worden. Hetzelfde geldt voor de beheerder van deze testomgeving en de testers die er op moeten gaan testen.

Voorbeeld

In figuur 8.9 “Het gebruik van de PFT-omgeving” staat het gebruik van de productiefixt est(PFT)omgeving toegelicht. De situatie is als volgt: in productie draait versie 1.0 van een systeem. Op hetzelfde moment wordt er gewerkt aan versie 1.1 in ontwikkeling en deze wordt al getest in de systeemtestomgeving. Dan treedt de productieverstorende fout op en deze moet opgelost worden. Dit gebeurt in de ontwikkelomgeving waar versie 1.01 ontstaat. Deze versie wordt geïnstalleerd in de productiefixtestomgeving waar deze getest wordt. Deze fix voor 1.0 moet ook worden geïmplementeerd in versie 1.1 (anders kan bij het gebruik van 1.1 de fout weer optreden). Dit betekent dat na de systeemtest er een nieuwe versie gemaakt moet worden van 1.1. Dit wordt 1.11 en deze versie kan weer naar de systeemtestomgeving en verder.



Figuur 8.9: het gebruik van de PFT-omgeving.

Uit dit voorbeeld wordt ook duidelijk hoe productieverstorende fouten kunnen doorwerken naar de rest van de organisatie. Wanneer systemen voor veel productieverstoringen zorgen, worden de werkzaamheden voor nieuwe versies steeds gestoord. Dit kan zelfs zo ver gaan dat nieuwe versies helemaal niet meer af komen, omdat ze steeds aangepast moeten worden aan de nieuwe fixes.

8.4.8 Testomgevingen bij uitbesteding

Wanneer een organisatie overgaat tot uitbesteding van testen (alleen of als onderdeel van een groter geheel) dan verdient het onderwerp testomgeving speciale aandacht. Zo kan er voor gekozen worden om alle activiteiten rond de testomgeving te laten uitvoeren door de leverancier (de “supply” organisatie). In het geval van complexe omgevingen met veel interfaces naar externe systemen kan de voorkeur uitgaan naar beheer door de uitbestedende organisatie (“demand” organisatie). Alleen dan moet de testomgeving wel benaderbaar zijn door de testers van de leverancier wat ook weer de nodige aandachtspunten oplevert. Ook is niet onbelangrijk welke partij de kosten van de infrastructuur voor haar rekening neemt. Vaak is dit de leverancier en worden de kosten indirect doorberekend aan de klant.

Er zijn in de regel diverse aspecten waarover een goed onderbouwde keuze moet worden gemaakt. Voorbeelden van deze aspecten zijn:

- toegang tot de omgeving: afspraken over hoe de testomgeving benaderd kan worden en door wie, bijvoorbeeld door een remote verbinding;
- inrichting en representativiteit van de testomgeving, met name bij structurele uitbesteding;
- de beschikbaarheid van de omgeving;
- actueel houden van de testomgeving: afspraken over het informeren en volgen van vereiste upgrades, vereiste configuraties, DBMS’en, specifiek besturingssysteem en middleware versies, licentiekosten;
- afspraken hoe om te gaan met beperkingen van de testomgeving die ervaren worden door een van de partijen.

8.4.9 Inrichten en beheren van testomgevingen als dienst

Een ontwikkeling van de laatste jaren is dat applicaties steeds meer gebruik maken van verschillende typen hardware en software. Bij veel (grote) organisaties bestaat de situatie dat verschillende hard- en software ook nog eens door verschillende partijen beheerd worden. Het opzetten van een testomgeving is hierdoor een organisatorische uitdaging. Er dienen erg veel contacten gelegd en afspraken gemaakt te worden. De situatie kan ontstaan dat problemen blijven 'hangen' tussen partijen: hardwareconfiguratie 1 is actief, hardwareconfiguratie 2 is actief en de koppeling ertussen is actief, maar berichten komen niet aan en geen van de drie (of meer!) beheerpartijen voelt zich eigenaar van het knelpunt.

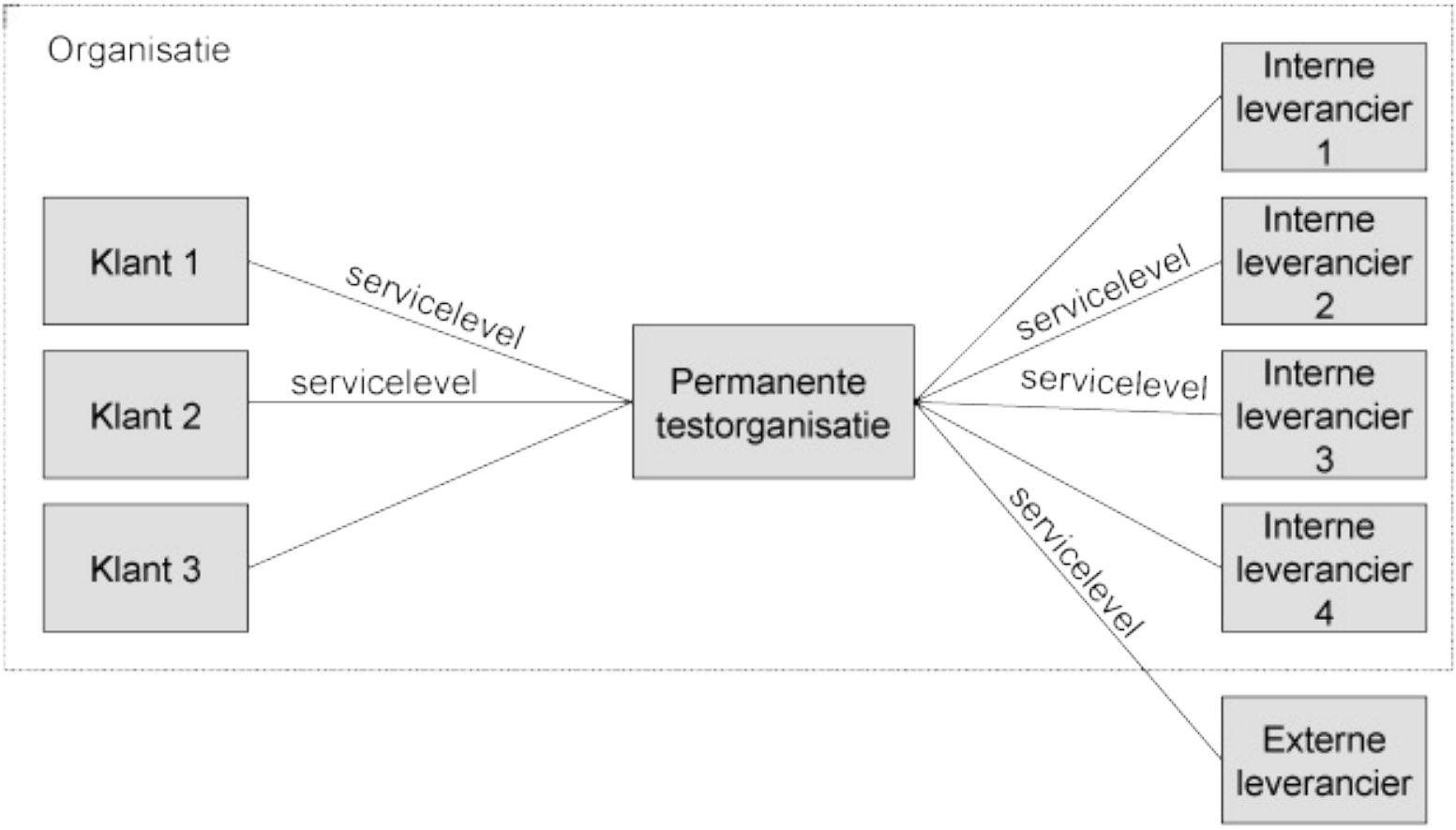
Samenvattend: de beschikbaarheid van (ketens van) testomgevingen is regelmatig een bron van problemen, zowel op het gebied van beschikbaarheid als consistentie. Als gevolg hiervan zien automatiseringstrajecten zich voor de keuze gesteld, of onvoldoende getest in productie te gaan, of de datum van in-productie-name uit te stellen. De slechte beschikbaarheid van testomgevingen vindt zijn oorzaak in de complexiteit, zowel organisatorisch als technisch. Er heerst een versnippering van taken en bevoegdheden tussen de gebruiker (tester, testmanager) en leveranciers van (onderdelen van) de testketen. Bovendien krijgt het oplossen van een niet-werkende testomgeving binnen organisaties vaak een lage prioriteit.

Organisaties kunnen er voor kiezen om dit knelpunt van de slechte beschikbaarheid op te lossen door de verantwoordelijkheid ervan in de lijn te beleggen. Een kandidaat hiervoor is de permanente testorganisatie die in paragraaf 8.3 "Permanente testorganisatie" staat beschreven. Deze permanente testorganisatie kan diensten leveren op het vlak van inrichten en beheren van (ketens van) testomgevingen.

In de volgende paragrafen wordt dieper ingegaan op het inrichten en beheren van testomgevingen als dienst. Als eerste wordt bepaald wat de rol kan zijn van een permanente testorganisatie bij het inrichten en beheren van testomgeving. Daarna zal gekeken worden naar de processen die binnen de testorganisatie ingericht moeten worden. De volgende stap is om te identificeren wat voor soorten omgevingen aangeboden kunnen worden en wat deze omgevingen uniek maakt.

De regisseursrol van de testorganisatie

Een permanente testorganisatie kan de rol van regisseur op zich nemen bij het inrichten en beheren van testomgevingen. Hierdoor dient het inrichten en beheren van testomgevingen goedkoper en kwalitatief beter te worden. Als regisseur is de testorganisatie het single point of contact voor de klanten van testomgevingen. Het is een vast aanspreekpunt voor de klant. De klant hoeft niet zelf meer problemen rondom testomgevingen op te lossen. De testorganisatie draagt zorg voor alle interne en externe contacten met de verschillende leveranciers van de (componenten van) testomgevingen. Rond de diensten die worden afgenomen van deze leveranciers kunnen servicelevels worden afgesproken en dat kan ook rond de diensten die worden aangeboden door de testorganisatie (zie figuur 8.10 "Regisseursol van de permanente testorganisatie").



Figuur 8.10: Regisseursrol van de permanente testorganisatie.

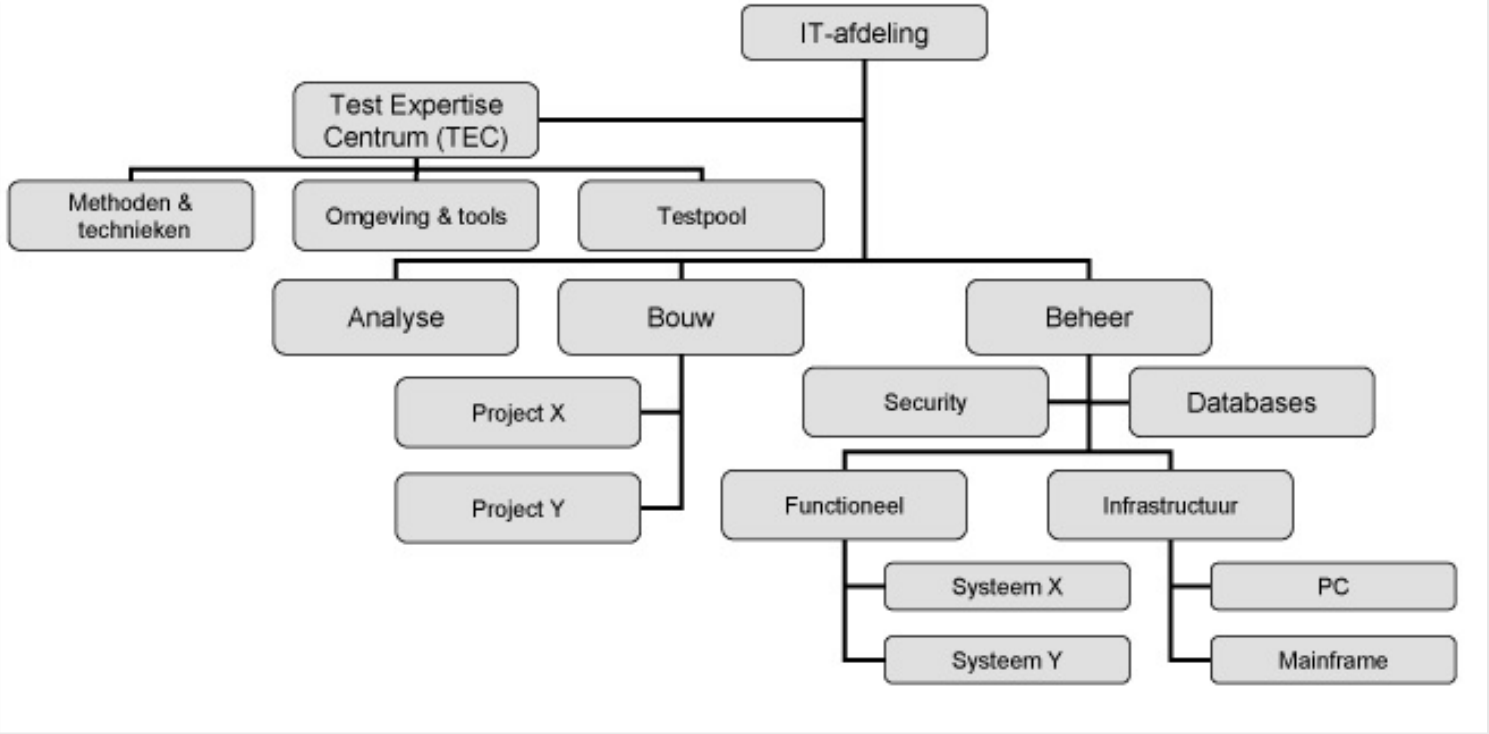
Wanneer er een incident (probleem[3]) in de testomgeving optreedt, dan zal de testorganisatie deze moeten (laten) oplossen. Per definitie is de testorganisatie de initiële eigenaar van dit incident totdat duidelijk is dat de oorzaak elders ligt. Door de regie te nemen in de verschillende processen (tot de productieomgeving) kunnen incidenten voorkomen worden wanneer andere partijen omgevingen gaan wijzigen. Binnen de testorganisatie dient een compleet beeld te zijn van de beschikbare componenten verdeeld over de verschillende hard- en software configuraties en de mogelijke koppelingen hiertussen. Hierdoor zal het eenvoudiger worden om oorzaken van incidenten te achterhalen en op te lossen.

De diensten die de testorganisatie biedt, moeten gestandaardiseerd worden. Alleen dan komen de voordelen van het centraal beheren en beschikbaar stellen van testomgevingen tot zijn recht. Het beschikbaar hebben van standaard diensten vertaalt zich naar het aanbieden van een vaste set aan mogelijke testomgevingen. Dit betekent dat de vraag van de klant altijd vertaald moet worden naar één van deze mogelijke testomgevingen. Hierin spelen de medewerkers van de testorganisatie een adviserende rol.

Voorwaarde is dat de medewerkers die deze dienst verzorgen inhoudelijk technische kennis van de testomgevingen moeten hebben. De organisatie rond deze dienst is ook veel meer een beheerorganisatie dan een testorganisatie. Dit betekent dat het profiel van de medewerkers ook meer richting beheer moet gaan dan richting testen.

Voorbeeld

In figuur 8.11 “Mogelijk organigram waarin een Test Expertise Centrum (TEC) de regiefunctie heeft voor het inrichten en beheren van testomgevingen” staat een voorbeeld van een organigram waarin een Test Expertise Centrum (TEC) bestaat zoals dat in paragraaf 8.3.7 “Test Expertise Centrum (TEC)” is uitgelegd. Binnen dit TEC bestaat een afdeling genaamd “omgeving en tools”. Deze afdeling vervult de regiefunctie voor het inrichten en beheren van testomgevingen. De leveranciers van de testomgevingen bestaan uit vele partijen. Zo bestaat er binnen beheer de afdeling infrastructuur. Die is georganiseerd naar de twee soorten hardwareconfiguraties die ze in beheer hebben (PC en mainframe). Voor de testomgeving zijn zij verantwoordelijk voor het leveren van de hardware, de netwerkcomponenten, de operating systemen enzovoort. Daarnaast bestaan er twee stafafdelingen binnen beheer; security en databases. De eerste stafafdeling regelt de aanvragen voor de verschillende autorisaties op logisch niveau op de gebruikte systemen. De andere stafafdeling, databases, is verantwoordelijk voor het beheer van de verschillende databases. Als laatste is er nog het functioneel beheer die de software beheert. Wanneer project X of project Y een testomgeving wil hebben dan moet zij met een aantal (en in het slechtste geval allemaal) partijen contact opnemen. Zij zijn dus de klanten van het TEC.



Figuur 8.11: Mogelijk organigram waarin een Test Expertise Centrum (TEC) de regiefunctie heeft voor het inrichten en beheren van testomgevingen.

Benodigde processen

Zoals in paragraaf 8.3.5 “Algemene procesmodel” beschreven, moet binnen de permanente testorganisatie een aantal processen zijn ingericht voor het aanbieden van de diensten. Dit geldt ook voor de situatie waarin de testorganisatie de rol van regisseur vervult bij testomgevingen. Er zijn twee belangrijke dienstenprocessen te onderscheiden:

- Inrichten van een nieuwe testomgeving;
- Beheren van een testomgeving.

Inrichten van een nieuwe testomgeving

Nadat het eerste contact met een klant is gelegd, zal tijdens de initiatiefase van de opdracht een intake plaatsvinden. Basis hiervoor is de definitie van de gewenste testomgeving die is opgesteld in het (master)testplan van de klant. Hierin staat de eerste inventarisatie van de eisen en wensen van de klant. De volgende stap, in de initiatiefase, is de verdere specificatie van de testomgeving. Hierbij wordt ondersteuning verleend door één of meer omgevingsspecialisten van de testorganisatie. De klant dient de specificatie van de testomgeving aan te leveren (bijvoorbeeld als een architectuurplaat). Door de specialist moet aangegeven worden wat wel en wat niet gerealiseerd kan worden en hoe dit binnen de standaarddiensten van de testorganisatie past. Vanaf het ogenblik dat de klant en de testorganisatie overeenstemming hebben over het plan met bijbehorende tijdslijnen en kosten (formeel startpunt) wordt er een opdrachtomschrijving opgesteld met daarin de beschrijving van de gewenste inrichting. Dan wordt er gestart met de uitvoering van de opdracht: het realiseren van de betreffende testomgeving. De omgevingsspecialisten gaan aan de hand van de opdrachtbeschrijving de complete testomgeving opzetten. Afhankelijk van de doorlooptijd zullen de specialisten statusrapportages aan de testorganisatie opleveren en deze zal op zijn beurt conform afgesproken tijdlijnen rapporteren aan de klant.

Beheren van een testomgeving

In de testorganisatie moet voor het beheren van de testomgevingen een aantal processen worden ingericht. Dit zijn:

Proces	Toelichting
Configuratiebeheer	Beschreven in paragraaf 8.4.6 “Processen bij testomgevingen”
Wijzigingenbeheer	Beschreven in paragraaf 8.4.6 “Processen bij testomgevingen”
Releasebeheer	Beschreven in paragraaf 8.4.6 “Processen bij testomgevingen”
Incidentbeheer	Het proces dat verantwoordelijk is voor het zo snel mogelijk oplossen van bevindingen op de testomgeving tijdens de test. Hieronder is ook bevindingenbeheer te scharen: het afhandelen van bevindingen tijdens het testen. Dit hoeven niet alleen bevindingen op de testomgeving te zijn, maar ook bevindingen die leiden tot een wijziging in de software of hardware. Belangrijk onderdeel bij dit proces is de inrichting van een service desk. Deze wordt gebruikt voor het afhandelen van vragen

	en klachten van klanten over het gebruik van de testomgeving.
Probleembeheer	Waar incidentbeheer verantwoordelijk is voor het zo snel mogelijk oplossen van een storing, houdt probleembeheer zich bezig met het zoeken van een structurele oorzaak van storingen en bevindingen. Hiervoor is vaak meer gedetailleerd onderzoek nodig. Dit gebeurt in overleg met ontwikkelaars en technisch en functioneel beheerders.
Gegevensbeheer	Elke test en elke omgeving stelt eisen aan de beschikbaarheid, omvang en inhoud (juistheid, compleetheid en actualiteit) van de gegevens. Dit kunnen testgegevens zijn, maar ook benodigde gebruikersprofielen of netwerkadressen. Op het moment dat de verschillende soorten testen worden uitgevoerd, dienen de juiste gegevens beschikbaar te zijn. Dit proces is daar verantwoordelijk voor. Hieronder valt ook het uitvoeren van back-ups en restores.
Operationeel beheer	Dit proces is verantwoordelijk voor het operationeel houden en monitoren van de verschillende componenten in de testomgeving. Dit gebeurt conform de eisen die met de klant zijn afgesproken.

De verschillende afdelingen die aan de permanente testorganisatie de omgevingen leveren zullen naar alle waarschijnlijkheid al werken volgens ITIL. Wanneer dit het geval is, is het aan te raden om voor bovenstaande onderwerpen gebruik te maken van al bestaande procedures, processen en aanwezige producten binnen die afdelingen. Zo zou de permanente testorganisatie gebruik kunnen maken van het servicedesk tool dat een eventuele IT helpdesk gebruikt.

Uitgediept

Verschillende belangen

Een potentieel knelpunt bij het beheren van testomgevingen door een permanente testorganisatie is het belangenverschil. Een testproject heeft een ander belang dan de testorganisatie. Het testproject wordt afgerekend op een projectresultaat en vaak geldt dat de deadline heilig is. De beheerder van de testomgevingen wordt op andere aspecten afgerekend. Vaak zijn dat onderwerpen als de kwaliteit van de omgeving, robuustheid, beschikbaarheid, consistentie van (test)gegevens of de snelheid waarmee een incident wordt afgehandeld. Deze zijn dan vertaald in zogenaamde KPI's (zie ook paragraaf 8.3.5 "Algemene procesmodel"). Het kunnen werken volgens een afrekenmodel dat gebaseerd is op KPI's vraagt om formele processen. In situaties waar het testproject vraagt om snel handelen (bijvoorbeeld bij het naderen van de deadline) kan deze formele manier van werken door de testorganisatie als "bureaucratisch" overkomen. Het is van belang dat beide partijen het belangenverschil onderkennen en er in het geval van (dreigende) knelpunten overlegd wordt over een oplossing.

Identificeren van testomgevingen met service levels

Wat maakt een systeemtestomgeving een systeemtestomgeving? Op het eerste gezicht lijkt dit een irrelevante vraag, maar in het licht van de regierol van de testorganisatie een zeer belangrijke. Want wat betekent het voor de klant wanneer de organisatie de dienst aanbiedt "opzetten systeemtestomgeving"? Of wat wil de klant wanneer deze vraagt om een ketentestomgeving?

Zoals in paragraaf 8.3.4 "Leveren van testdiensten" is beschreven, kan een testorganisatie verantwoordelijk zijn voor het leveren van een dienst. De grootste verantwoordelijkheid is bij diensten met een resultaatverplichting. Hierbij moet de permanente testorganisatie op basis van de gestelde vraag een resultaat leveren. Deze levering dient te geschieden binnen een vooraf gesteld tijdbestek, tegen vooraf gestelde kosten en met een vooraf gesteld kwaliteitsniveau. De permanente testorganisatie is verantwoordelijk voor het garanderen van de continuïteit in deze resultaatslevering. De norm van deze verplichting wordt vastgelegd in zogenaamde service levels.

Wanneer een testorganisatie als dienst het leveren van een systeemtestomgeving heeft, dan kan deze dienst specifieker gemaakt worden door het gebruik van service levels. Het vastleggen van service levels voor testomgevingen gebeurt door middel van het invullen van specifieke kenmerken van de omgeving. Voorbeelden hiervan zijn:

- Doel van de omgeving en de testsoorten waar het voor gebruikt mag worden;
- De doelgroep van de testomgeving;
- Koppeling met andere systemen of gebruik van stubs en drivers;
- De testgegevens in de omgeving en de verversingfrequentie;
- De security-eisen die op de omgeving van toepassing zijn;
- De beschikbaarheid van de omgeving;
- Beschikbare tooling in de omgeving;
- Eventuele entry- en exitcriteria die gelden.

Voorbeeld

Een permanente testorganisatie biedt als dienst het opzetten en beheren van een systeemtestomgeving aan. Aan deze dienst hangen de volgende service levels:
Doel: gecontroleerde omgeving waarin klant valideert of de applicatie gebouwd is volgens specificatie
Testsoorten: systeemtest en functionele acceptatie test

Doelgroep: testers van de klant
Koppelingen: standaard stubs & drivers, op afroep echte koppelingen mogelijk
Testgegevens: geen testgegevens aanwezig, verantwoordelijkheid klant
Security: standaard set aangeleverd, verder beheer verantwoordelijkheid klant
Entry-criteria: geen
Exit-criteria: testgegevens verwijderd, geen restgegevens meer aanwezig **Tooling:** Tools voor manipulatie database en tijdreizen
Beschikbaarheid: tijdens kantooruren (09.00 – 17.00) een 80% beschikbaarheid met een maximale downtime van 2 uur.

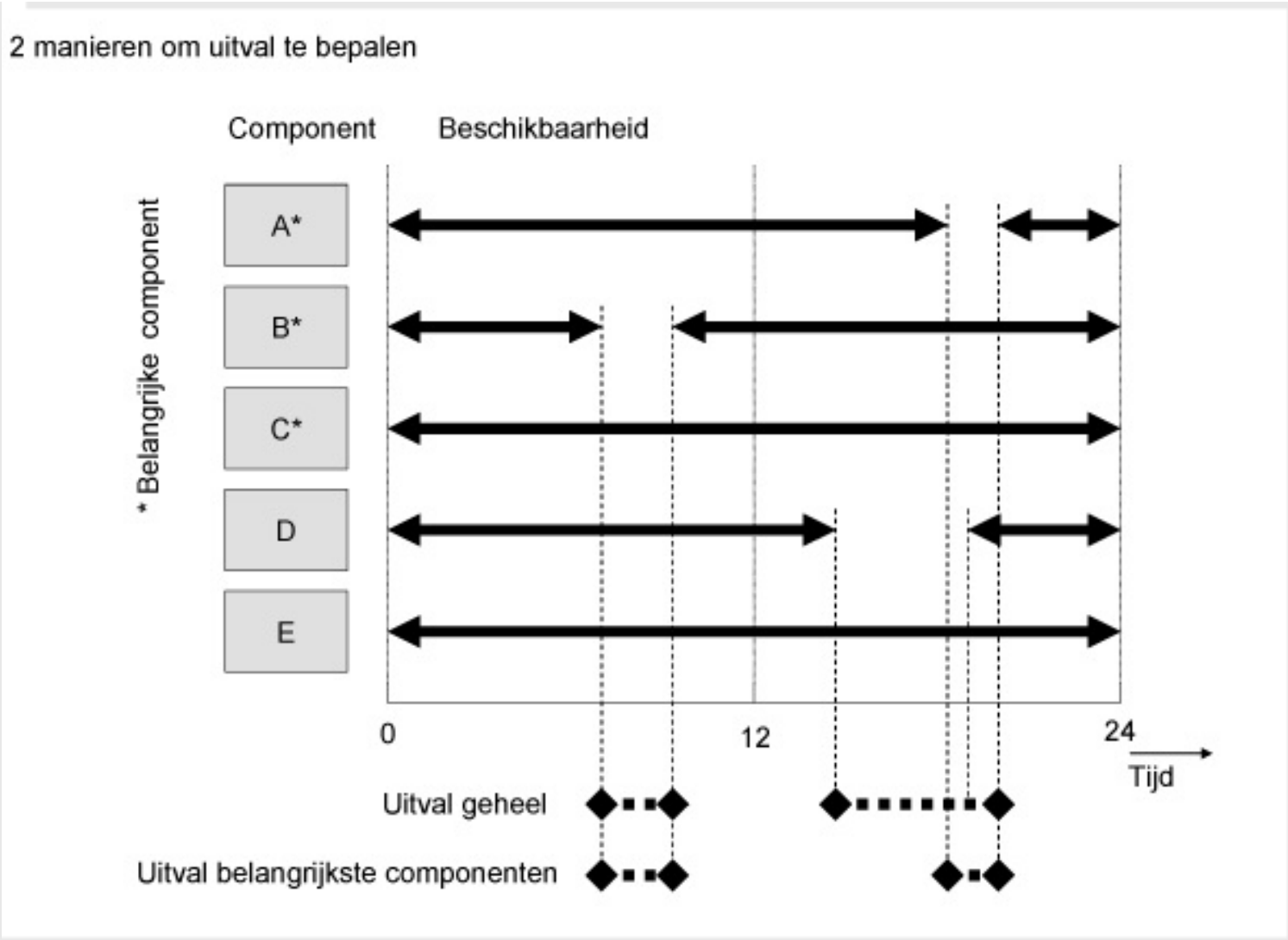
Uitgediept

Beschikbaarheid van testomgevingen

Wanneer de beschikbaarheid van de testomgeving genoemd wordt in een service level, dan is het belangrijk af te spreken hoe deze beschikbaarheid wordt gemeten. Ofwel, wat wordt er door beide partijen verstaan onder het beschikbaar zijn van de testomgeving. Dit kan namelijk verschillend zijn. Zo is het voor de ene partij voldoende dat de belangrijkste componenten beschikbaar zijn terwijl een andere partij het van belang vindt dat alles beschikbaar is. Deze twee verschillende invullingen geven wel een verschillend resultaat in figuur 8.12 “Schematisch overzicht van de beschikbaarheid van verschillende componenten in een testomgeving”. In dit schema bestaat een testomgeving uit 5 componenten (A, B, C, D en E). A, B, en C worden geïdentificeerd als de belangrijkste componenten. Dit zijn bijvoorbeeld componenten waarvan de correcte werking van het testobject direct afhankelijk is. Voorbeelden zijn databaseserver, web-server of berichtenbussen. Componenten D en E worden minder belangrijk gevonden. Dit zijn bijvoorbeeld koppelingen met andere systemen.

Van ieder van deze componenten wordt de beschikbaarheid gemeten. Zo zijn C en E continu beschikbaar, maar zijn er bij de andere 3 wat storingen te zien (onderbreking van de lijn). Wanneer er gekeken wordt naar alle componenten samen dan is het totaal van de storingen groter dan wanneer er alleen gekeken wordt naar de belangrijkste componenten.

Dit verschil van beschikbaarheid is vooral van belang bij verschillende testsoorten. Bij een systeemtest is het bijvoorbeeld minder belangrijk dat de koppelingen continu beschikbaar zijn dan bij een ketentest.



Figuur 8.12: Schematisch overzicht van de beschikbaarheid van verschillende componenten in een testomgeving.

8.5 Testtools

8.5.1 Inleiding

De ontwikkeling van de laatste jaren die kernachtig kan worden samengevat met "*meer voor minder en sneller en beter*", heeft een impact op alle disciplines binnen de IT (zie ook paragraaf 8.1 "Inleiding"). Met zeer geavanceerde ontwikkelomgevingen kunnen ontwikkelaars relatief eenvoudig en snel complexe programma's ontwerpen en bouwen. De iteratieve ontwikkelmethoden, die uitgaan van vergaande interactie met de gebruikers, zorgen er onder andere voor dat bouwtrajecten sneller tussentijds opleveren. Deze tussentijdse opleveringen worden dan getoetst aan de eisen en wensen van de gebruikers en de bevindingen worden verwerkt in de software. Dit betekent dat de software continu wijzigt en er voortdurend regressierisico's worden gelopen. Daarnaast wordt steeds meer gestuurd op hergebruik van interne en externe componenten die geïntegreerd moeten worden in de bestaande IT-architecturen. Dit heeft de benodigde tijd voor het ontwikkelen van nieuwe systemen bekort en het testen komt hiermee, ten opzichte van ontwikkeling en onderhoud, nog nadrukkelijker op het kritieke pad te liggen. Het dreigt zelfs een belemmerende factor te worden.

Al deze factoren, samen met het gegeven dat het testen van systemen nu al wordt gezien als een tijdrovende en kostbare activiteit, maken een hogere productiviteit van de testers en een hogere kwaliteit van de test noodzakelijk. Hierbij kunnen testtools als instrument worden gebruikt om dit te bereiken.

Het beschikbaar stellen van testtools aan testers wordt vaak bij een aparte afdeling belegd. Een reden hiervoor is het feit, dat het inrichten en beheren van testtools een specifieke expertise is. Het is iets waar testers over het algemeen weinig kennis van hebben. Een andere reden voor het beleggen van testtools bij een aparte afdeling is dat er vaak grote investeringen gemoeid zijn met de introductie van tools in een organisatie. Naast de hoge aanschafkosten moet er geïnvesteerd worden in het opleiden van de mensen en het ontwikkelen van nieuwe procedures. Er is dus tijd nodig om deze investering terug te verdienen en vaak is deze terugverdientijd langer dan één project.

In deze paragraaf wordt dieper ingegaan op testtools en het gebruik daarvan. In paragraaf 8.5.2 “Testtools toegelicht” wordt toegelicht wat een testtool is. In de paragraaf daarna (8.5.3 “Soorten testtools”) worden de verschillende soorten testtools beschreven. Paragraaf 8.5.4 “Voordelen gebruik testtools” beschrijft de voordelen van het gebruik van testtools. De laatst paragrafen beschrijven hoe testtools op basis van een toolbeleid ingevoerd kunnen worden bij testorganisaties. Hiervoor wordt eerst in paragraaf 8.5.5 “Invoeren van testtools met toolbeleid” het begrip toolbeleid geïntroduceerd en het faseringsmodel beschreven. Daarna komen achtereenvolgens de drie fasen aan bod, te weten de fase Initiatie (8.5.6), de fase Realisatie (8.5.7) en als laatste de fase Exploitatie (8.5.8).

8.5.2 Testtools toegelicht

Definitie

Een testtool is een geautomatiseerd hulpmiddel dat ondersteuning biedt aan één of meer testactiviteiten, zoals plannen, beheren, specificeren en uitvoeren.

Eén van de voorwaarden voor succesvol gebruik van testtools is de aanwezigheid van een gestructureerde testaanpak. In een goed beheerst proces kunnen tools zeker een belangrijke meerwaarde geven, maar bij een onvoldoende beheerst testproces werken ze contraproductief. In feite automatiseren testtools het testproces en dit vraagt om een zekere herhaalbaarheid en standaardisatie van de te automatiseren activiteiten. Een ongestructureerd proces kan niet aan deze voorwaarden voldoen. De inzet van testtools kan wel als hefboom fungeren om een gestructureerde aanpak te implementeren. Structurering en automatisering moeten dan echter minimaal hand in hand gaan.

Uitgediept

Terminologie: tools, testtools en CAST tools

Er wordt op verschillende wijzen gesproken over tools die binnen een testproces worden gebruikt. Zo kan men spreken over gewoon *tools* maar andere voorbeelden zijn *testtools*, *CAST tools* (CAST staat voor Computer Aided Software Testing) of *testautomatisering*. Het is niet mogelijk een eenduidige keuze te maken voor de juiste terminologie. Er zijn partijen die stellen dat een tool een testtool is wanneer deze alleen gebruikt kan worden ter ondersteuning van een specifieke testactiviteit. Het tegenargument is dan dat sommige testtools, die dienen ter ondersteuning van de testuitvoering, soms worden gebruikt bij andere activiteiten. Een voorbeeld is een testtool waarmee de testuitvoering geautomatiseerd kan worden. Die tool werkt op basis van het automatiseren van handelingen en die kan ook gebruikt worden voor dataconversie. En daarmee is het weer een tool. Binnen TMap worden de termen tools en testtools door elkaar gebruikt.

Het kunnen gebruiken van testtools mag tegenwoordig als basisvaardigheid van een tester verondersteld worden. Het kunnen inrichten en beheren van testtools alsmede het vergaand automatiseren van routinematige activiteiten (bijvoorbeeld de testuitvoering) vereist daarentegen een specialistische en diepgaande kennis van programmeren en tools. Deze kennis is zeker niet bij iedere tester aanwezig. Zodoende is een nieuw soort specialisme ontstaan: testtoolprogrammeur, testtoolspecialist en testtoolconsultant. In paragraaf 8.6 “Testprofessionals” worden deze functies verder beschreven.

Uitgediept

Prijsstructuur van testtools

Testtools bestaan in alle vormen en maten. Deze tools kennen alle hun eigen prijsstructuur. Commerciële tools kennen vaak een licentiesysteem waar, op basis van het aantal gebruikers van de tool, een eenmalige prijs wordt afgesproken. Naast deze eenmalige prijs wordt dan een jaarlijkse contract afgesloten waardoor de organisatie zich verzekert van ondersteuning door de leverancier van de tool en het verkrijgen van de nieuwe updates en releases. Dit heet vaak een onderhouds- of servicecontract.

Andere varianten zijn testtools met een prijsstructuur op basis van de varianten *shareware*, *freeware* en *open-sourcesoftware*. Shareware is een prijsstructuur van software die zonder of met weinig restricties verspreid mag

worden, maar waarvoor bij herhaald gebruik wel een vastgelegd bedrag betaald moet worden. Freeware is software waarvan de auteur een licentie heeft verleend tot gebruik en verdere verspreiding in ongewijzigde vorm, zonder daarvoor vergoeding te vragen. Open-sourcesoftware gaat nog een stap verder dan freeware. De auteur geeft, naast het vrij verspreiden van de software, ook toestemming de software aan te passen. Deze aangepaste software mag ook weer vrij verspreid worden.

In tegenstelling tot open-sourcesoftware wordt freeware volledig beschermd door auteursrechten. En in tegenstelling tot open-sourcesoftware wordt de broncode van freeware over het algemeen niet beschikbaar gesteld. Steeds meer (zelfgemaakte) testtools worden op basis van deze varianten door de maker beschikbaar gesteld via de verschillende mogelijkheden van internet.

8.5.3 Soorten testtools

Testtools bieden ondersteuning bij het uitvoeren van bepaalde activiteiten in de verschillende fasen van TMap. Er bestaan verschillende soorten testtools en deze kunnen in vier groepen worden ingedeeld:

1. tools voor het plannen en beheren van de test;
2. tools voor het ontwerpen van de test;
3. tools voor het uitvoeren van de test;
4. tools voor het debuggen en analyseren van de code.

De testtools in groep 1, 2 en 3 worden voornamelijk gebruikt door de testers in het onafhankelijke testteam. Van deze groepen volgt hieronder, per groep, een toelichting en een overzicht van diverse beschikbare soorten testtools.

De testtools in groep 4 worden voornamelijk gebruikt door de ontwikkelaar.

Deze worden in hoofdstuk 7 “Ontwikkeltesten” beschreven. Op www.tmap. net staat de matrix “Tools per TMap activiteit” met daarin een overzicht van welke soort testtool bij welke activiteit van TMap hoort.

Tools voor het plannen en beheren van de test

Net zoals een bedrijfsproces ondersteund kan worden door geautomatiseerde hulpmiddelen, kan ook een testproces ondersteund worden door geautomatiseerde hulpmiddelen. Dit zijn testtools die activiteiten rond het plannen en beheren van de test ondersteunen, zoals het opstellen van de planning, bewaken van de voortgang en het registreren van bevindingen. Omdat de tools zich richten op het proces, werken ze technisch onafhankelijk van het testobject. De volgende soorten tools behoren tot deze groep:

- Testwarebeheertool;
- Bevindingenbeheertool;
- Plannings- en voortgangsbewakingstool;
- Workflowtool.

Uitgediept

Testmanagementtool geen aparte toolsoort

Binnen TMap is een testmanagementtool niet benoemd als een aparte toolsoort. De reden hiervoor is dat het een geïntegreerde set aan functionaliteiten biedt op het gebied van verschillende toolsoorten. Zo ondersteunt een testmanagementtool vaak testwarebeheer, bevindingenbeheer en planning en voortgangsbewaking. Hoewel de functionaliteit voor elk gebied meestal niet zo uitgebreid is als bij een specifieke toolsoort, ligt de kracht van een testmanagementtool in de integratie van de diverse tools. Vaak zijn de testmanagementtools ook geïntegreerd met tools voor geautomatiseerde testuitvoering. De testmanagementtool kan ook een geautomatiseerde workflow bevatten. Hierdoor ondersteunt het gebruik van de tool het hele testproces; van het maken van het testplan tot en met het rapporteren over de resultaten.

Testwarebeheertool

Tijdens het testproces ontstaan allerlei producten die gezamenlijk de testware vormen. Het is van groot belang dat er tijdens een testproces voor wordt gezorgd dat de producten adequaat worden beheerd. Testwarebeheertools ondersteunen het registreren van de verschillende versies van testware die gedurende het testproces ontstaan en van de mogelijke relaties tussen deze testware. Zo kan bijvoorbeeld worden afgeleid welk testresultaat bij welke versie van het testscripts hoort, of welke versie van de testspecificatie hoort bij welke versie van de testbasis. Daarnaast dwingt testwarebeheer een zekere mate van structuur en eenduidigheid af.

Bevindingenbeheertool

Deze tools ondersteunen het registreren en afhandelen van testbevindingen die tijdens een testproces worden gedaan. Het proces van bevindingenbeheer is complex en omvangrijk. Het aantal testbevindingen bedraagt, onder andere afhankelijk van de omvang en kwaliteit van het testobject, soms honderden of zelfs duizenden. Bevindingen kunnen verder één of meer bijlagen bevatten met daarin schermprints of delen van de testbasis ter verduidelijking. Bij de afhandeling van testbevindingen zijn meerdere partijen betrokken die vaak ook op verschillende locaties zitten. Soms is de procedure voor het afhandelen van bevindingen afhankelijk van de urgentie van de bevinding. Ter ondersteuning hiervan zijn tools beschikbaar. Naast de registratie, kan ook de levenscyclus van een bevinding worden gevolgd en bewaakt. Sommige tools voorzien tevens in de mogelijkheid van het creëren van managementrapportages en metrics.

Plannings- en voortgangsbewakingstool

Bij omvangrijke testprocessen is een tool ter ondersteuning van het proces van planning en voortgangsbewaking onmisbaar. Een planning moet in zijn geheel worden doorberekend voor wat betreft activiteitsduur, start- en eventuele einddata en toegekende middelen. Veelal voorzien planningspakketten in “what if”-analyses en zijn deze in staat om zowel strokenplanningen als netwerkplanningen te genereren. Dit soort tools helpen bij het begroten en plannen van de test en op www.tmap.net staat hier een voorbeeld van. Voortgangsbewaking dient inzicht te geven in de gemaakte voortgang en hier moeten rapportages van gemaakt kunnen worden. Daarnaast zal het inzicht moeten kunnen geven in de nog benodigde tijd en resources om het testproces af te ronden. Een belangrijk aspect bij de selectie van tools voor planning en voortgangsbewaking is de mogelijkheid tot het vervaardigen van managementinformatie, bijvoorbeeld overzichten van resources en kosten.

Workflowtool

Het testproces van TMap kent verschillende fasen met activiteiten en deelactiviteiten. Een aantal van deze activiteiten is afhankelijk van elkaar. De output van een activiteit is dan de input van een andere activiteit en zo ontstaan meerdere ketens van activiteiten (workflow). De activiteiten binnen een keten worden door één of meer personen binnen het testteam uitgevoerd. Het managen van dit gehele proces met de verschillende ketens van activiteiten is bij grote testteams complex. Een workflowtool kan hier ondersteuning bij bieden. De workflowtool kent de uit te voeren activiteiten en zorgt voor de routering van het werk naar de betrokken personen. De testmanager heeft met de tool continu inzicht in de status van de uit te voeren activiteiten en kent de totale werkvoorraad. De tool signaleert overschrijding in planningen, of van ongebruikelijke werkvoorraden, zodat de testmanager kan ingrijpen.

Tools voor het ontwerpen van de test

Tools die het specificeren van testgevallen ondersteunen of testgevallen zelfs geheel automatisch genereren behoren tot deze groep. Maar ook de testtools waarmee de testgegevens kunnen worden gecreëerd, opgezet en beheerd, behoren hier toe. Tools die het maken van testgevallen ondersteunen, doen dit meestal op basis van een basistechniek (zie paragraaf 14.2.3 “Testontwerptechniek en basistechniek). Wanneer de testbasis in formele notatie is beschreven kunnen deze testtools automatisch testgevallen genereren. In veel gevallen moeten deze testgevallen nog wel verder bewerkt worden. De tool verleent hierbij dan ondersteuning. De volgende soorten tools behoren tot deze groep:

- Testdatatool;
- Testontwerptool;
- ‘Model based testing’-tool.

Testdatatool

Deze tool helpt bij het (op)bouwen van fysieke sets van testgegevens. Met behulp van generatoren kan onder andere een

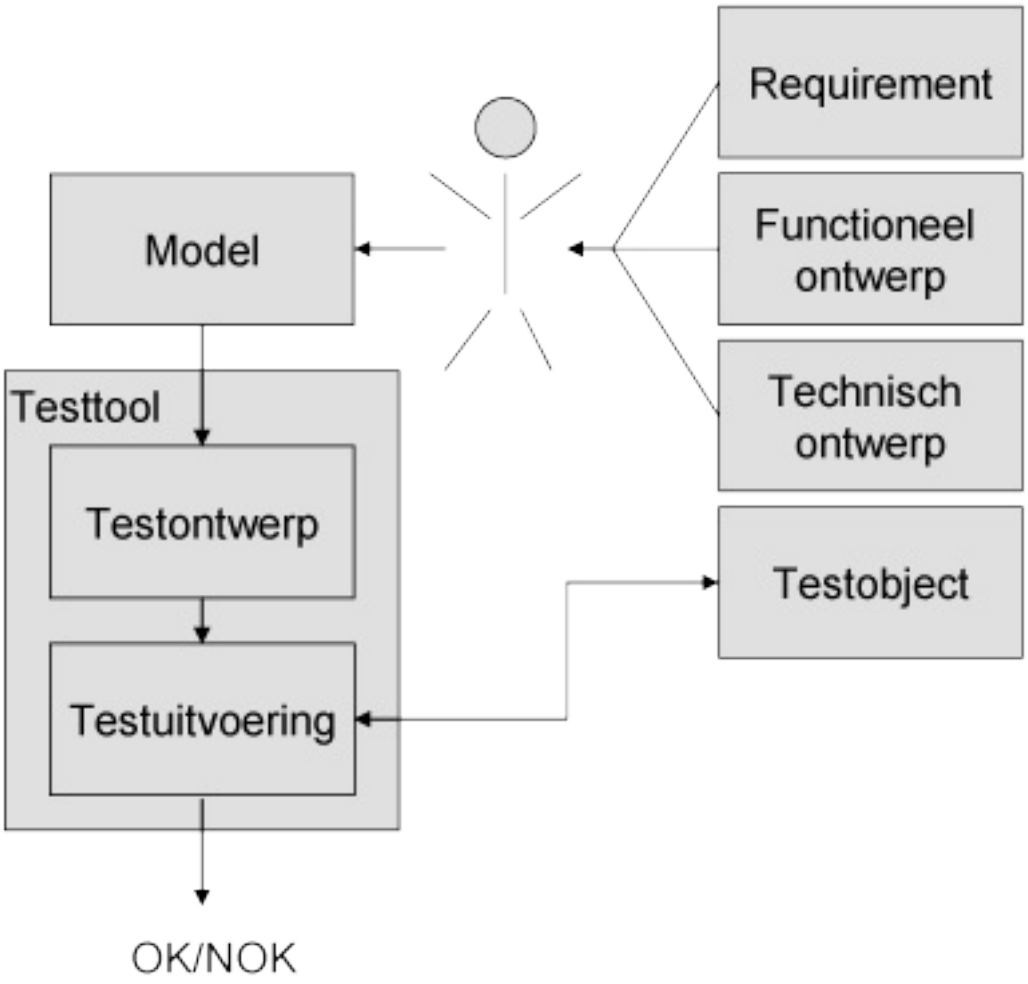
random invulling worden gegeven aan de hand van een bestands- en/of databasespecificatie. Op deze wijze kan relatief snel een omvangrijke set van testgegevens worden gecreëerd voor bijvoorbeeld een real-life test. De “regels” voor het genereren van testgegevens moeten vooraf gespecificeerd worden in de tool. Denk hierbij bijvoorbeeld aan het definiëren van begrensde verzamelingen waaruit geselecteerd mag worden en van relaties tussen verschillende gegevens (consistentieregels).

Testontwerptool

Wanneer tijdens de specificatie van testgevallen gebruik wordt gemaakt van testontwerptechnieken, dan bieden deze tools ondersteuning. Vooral wanneer bij testen gebruik wordt gemaakt van verschillende mogelijke combinaties van invoer laten deze tools snel hun toegevoegde waarde zien. Voor meer informatie over dit onderwerp zie paragraaf 14.3 “Dekkingsvormen en basistechnieken”.

‘Model based testing’-tool

Deze tools bieden ondersteuning bij de aanpak van Model Based Testing. Dit is een aanpak waarbij op basis van een model van het testobject, testgevallen worden ontworpen (Figuur 8.13: Model based testen). Deze testgevallen worden dan geautomatiseerd uitgevoerd op het testobject. De uitdaging bij deze aanpak is onder andere het opstellen van een formeel model waarin de werking van (een deel van) de applicatie wordt weergegeven. Het opstellen van dit model is mensenwerk. Wanneer dit model klaar is kan het ingelezen worden door een tool die de creatie en uitvoering van testgevallen voor zijn rekening neemt. Vooral bij (een combinatie van) complexe systemen die oneindig veel verschillende mogelijkheden kennen, loont het op deze manier te werk te gaan. Voor meer informatie over Model Based Testing zie www.model-based-testing.org.



Figuur 8.13: Model based testen.

Uitgediept

Tekstverwerking- en spreadsheetprogramma’s bezien als testtools

Er wordt wel eens gezegd dat de testtools die een tester het meest gebruikt de tekstverwerking- en spreadsheetprogramma's zijn. Op het eerste gezicht lijkt dit een rare opmerking. Maar wanneer verder wordt gekeken dan alleen de standaard functionaliteit van deze tools, dan kan hier toch een kern van waarheid in schuilen. Deze tools kunnen activiteiten van een tester ondersteunen en in sommige gevallen zelfs automatiseren. Door middel van het eenvoudig kopiëren en plakken van stukken tekst wordt het hergebruik bij het maken van testscripts vereenvoudigd. Het gebruik van een spreadsheet voor de notatie van de logische en fysieke testgevallen (in de verschillende cellen) dwingt een standaard manier van werken af. Dit komt de interpretatie door de verschillende testers ten goede. Daarnaast bevatten de meeste tekstverwerking- en spreadsheetprogramma's zogenaamde "macro"- functionaliteiten, waarmee handelingen in deze programma's geautomatiseerd kunnen worden. In sommige gevallen kunnen zelfs koppelingen worden gelegd met externe programma's. Hierdoor is het dan mogelijk om met een tekstverwerking- of spreadsheetprogramma een activiteit als testuitvoering (in zeer lichte vorm) te automatiseren.

Voorbeeld

Een batchsysteem maakt als invoer gebruik van tekstbestanden. Deze tekstbestanden bevatten regels met verschillende data. Iedere regel bestaat uit 10 of meer dataelementen en deze worden van elkaar gescheiden door een komma. Het testen van dit batchsysteem gebeurt door middel van deze tekstbestanden. Testers moeten dus hun testgevallen aanleveren in het formaat van de tekstbestanden. Het lezen en begrijpen van de inhoud van een tekstbestand is lastig. De betekenis van een dataelement is afhankelijk van de positie waar het staat en de waarde. Het maken van tekstbestanden om te testen is dus zeer complex. Er is gekozen om in een spreadsheetprogramma de tekstbestanden te maken. Door elke kolom te koppelen met een positie (en de betekenis) in het tekstbestand wordt het voor de testers eenvoudig het tekstbestand op te bouwen. Met een druk op de knop wordt dan, op basis van de verschillende cellen, een tekstbestand gemaakt voor het batchsysteem. Zie figuur 8.14 "van een spreadsheet voor de creatie van een tekstbestand" voor een schematische weergave.

Figuur 8.14: Gebruik van een spreadsheet voor de creatie van een tekstbestand.

Tools voor het uitvoeren van de test

Deze testtools worden ingezet op het kritieke pad van testen: het uitvoeren van testscripts. Omdat deze tools zich richten op het product, moeten ze technisch samenwerken met het testobject en de bijbehorende hard- en softwarecombinatie. Het inzetten van dit soort testtools loont als de testactiviteiten nauwkeurig moeten worden uitgevoerd en relatief routinematig zijn. Voorbeelden hiervan zijn het veelvuldig uitvoeren van dezelfde test en het vergelijken van

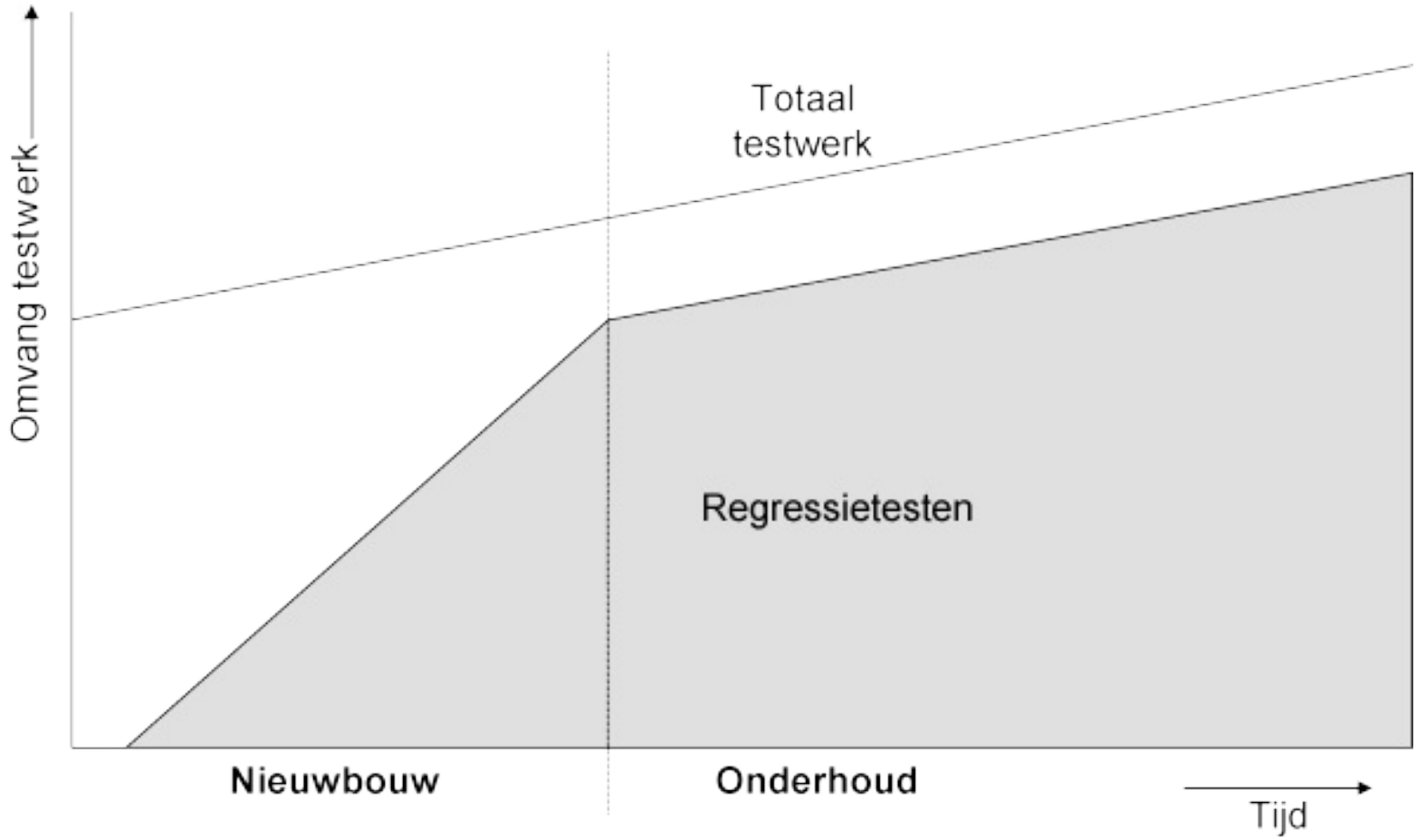
omvangrijke overzichten met als doel vast stellen of beide overzichten hetzelfde zijn. Maar ook activiteiten die veel technische kennis vereisen (bijvoorbeeld testen van beveiliging) of waar een grote hoeveelheid testers voor nodig zijn (bijvoorbeeld testen met load profiles) kunnen door deze testtools worden uitgevoerd.

De volgende soorten tools behoren tot deze groep:

- Geautomatiseerde testuitvoering tool;
- Performance-, load- en stresstesttool;
- Monitor;
- Code coveragetool;
- Comparator;
- Databasemanipulatietool;
- Simulator;
- Stubs en drivers.

Geautomatiseerde testuitvoering tool

De tools voor geautomatiseerde testuitvoering spreken veel organisaties tot de verbeelding. Want voor veel organisaties is het steeds weer opnieuw testen van ongewijzigde functionaliteit (regressietesten) het meest omvangrijke en tijdrovende deel in het testtraject. Regressietesten begint al tijdens de nieuwbouw van een systeem en krijgt gedurende de verdere levenscyclus een groeiend aandeel in de totale omvang van het testwerk (zie figuur 8.15 “Toenemend aandeel van regressietesten gedurende de levenscyclus”). Het geautomatiseerd uitvoeren van deze regressietests kan tijdwinst opleveren. Dit spreekt niet alleen de tester aan, die zijn steeds terugkerende en daarmee saaie dagelijkse activiteiten uit handen genomen ziet worden, maar ook de calculerende testmanager die tientallen procenten wil besparen.



Figuur 8.15: Toenemend aandeel van regressietesten gedurende de levenscyclus.

Van dit soort testtools bestaan twee varianten:

- Tools die de testuitvoering automatiseren via de gebruikersinterface (GUI) van de te testen applicatie. Dit noemt men ook wel record & playbacktool.
  Een record & playbacktool legt de testinvoer (gegevens en acties) en het verwachte resultaat vast in een script. Dit script kan door de tool op een later tijdstip opnieuw worden afgespeeld, zodat de test gemakkelijk kan worden herhaald

(de term “script” moet in deze context overigens niet verward worden met de handmatige testscripts die onderdeel zijn van de testspecificaties).
- Tools die de testuitvoering automatiseren via een programma-interface.
  Voorbeelden van een programma-interface zijn Application Programming Interface (API) of berichten in het formaat van XML. Vaak bieden dit soort tools de mogelijkheid om opgeslagen invoergegevens te muteren en ook leveren zij ondersteuning bij het aanmaken van testinvoer. In het algemeen worden deze tools gecombineerd met vergelijkingstools om het analyseren van de testresultaten mogelijk te maken.

Het grote voordeel van de tools voor geautomatiseerde testuitvoering is dat een test op een later moment geautomatiseerd kan worden herhaald. Dit voordeel zal teniet worden gedaan indien het testobject zodanig is gewijzigd dat het geautomatiseerde script blokkeert tijdens het afspelen. Voor een efficiënt gebruik van de tool is onderhoud aan de geautomatiseerde scripts vereist. Dit onderhoud mag niet meer kosten dan het voordeel wat de geautomatiseerde testuitvoering gaat opleveren. De wijzigingen in het testobject mogen maar een beperkt aantal aanpassingen in de geautomatiseerde scripts tot gevolg hebben. Bij regressietesten is dit vaak het geval zodat deze soort tools uitermate geschikt zijn voor deze testvorm.

Het geheel van tool, framework, testgevallen, scripts en vastgelegde resultaten wordt een testsuite genoemd. Onder een framework wordt een bibliotheek van herbruikbare geautomatiseerde scripts verstaan. Elk geautomatiseerd script is feitelijk een (klein) programma. Het hanteren van de basisprincipes van modulair programmeren vergroot de onderhoudbaarheid van deze scripts: elke groep opeenvolgende acties die herhaaldelijk moet worden uitgevoerd (bijvoorbeeld verplaatsing naar een bepaald scherm in de applicatie) kan het best als afzonderlijk script worden opgeslagen. Tests, die deze groepering van activiteiten moeten uitvoeren, roepen het betreffende script aan. Wanneer er iets verandert in de groep van activiteiten (bijvoorbeeld door een andere menu-opzet), hoeft slechts één script aangepast te worden. De geautomatiseerde scripts bestaan op verschillende abstractieniveaus. Dit kan variëren van het activeren of controleren van een specifiek object van het te testen systeem tot het uitvoeren van een bedrijfsproces. Om een testsuite op een dergelijke modulaire wijze te bouwen, is deskundigheid vereist op het gebied van testen en van softwareontwikkeling.

Het beschikken over een dergelijke testsuite geeft de mogelijkheid om in korte tijd nieuwe testsuites (voor nieuwe systemen) te bouwen, omdat een groot aantal van de benodigde bouwstenen (scripts) al aanwezig is in de bibliotheek. De investering in een testsuite moet zich terugverdienen in termen van sneller en beter opnieuw testen van nieuwe releases. De benodigde inspanning om de testsuite aan te passen voor een nieuwe release moet opwegen tegen de baten die het gebruik van deze testsuite met zich meebrengt. De belangrijkste kwaliteitseisen die aan een testsuite gesteld moeten worden, zijn dan: onderhoudbaar, flexibel, robuust en herbruikbaar. Om testsuites te ontwikkelen die aan deze eisen voldoen, wordt verwezen naar het boek Automatisering van de testuitvoering [Broekman, 2001].

Performance-, load- en stresstesttool

Performance-, load- en stresstesttools kunnen een informatiesysteem belasten door (grote aantallen) gebruikers te simuleren. Het doel van dit soort testen is het bepalen of het systeem correct en snel genoeg blijft functioneren onder de verwachte productiebelasting. Voor meer informatie hierover zie paragraaf 14.3.8. “Statistisch gebruik: Operational profiles en Load profiles” en paragraaf 10.3.3. “Performance”. Om bij de gemeten resultaten aan te kunnen geven waar de mogelijke oorzaken liggen van problemen zijn de tools vaak gecombineerd met monitoren.

Monitor

Om inzicht te krijgen in aspecten als geheugenbeslag, CPU-gebruik, netwerkbelasting en performance kan tijdens het testproces gebruik worden gemaakt van monitors. Allerlei gegevens betreffende het middelenbeslag worden gemeten en opgeslagen. De meetgegevens worden vervolgens door middel van een rapportage gepresenteerd. Het inregelen van dit soort tools is veelal een complexe aangelegenheid. Vaak echter zijn bij een beheerafdeling al monitoren aanwezig voor het bewaken van de operationele productieomgeving. Wellicht kunnen deze ook in de testomgeving worden gebruikt. Bij performance-, load- en stresstesttools is monitoringfunctionaliteit vaak een geïntegreerd onderdeel.

Code coveragetool

Code coveragetools leveren informatie over welke delen van de programmacode tijdens een test zijn doorlopen. Ze bieden daarmee nuttige ondersteuning voor het meten van het effect van de gebruikte testontwerptechnieken. De metingen worden verricht op programmaniveau of op subsysteemniveau.

Op deze manier wordt vastgesteld of bij het testen elk programmastatement tenminste eenmaal wordt uitgevoerd. De conclusies die hieruit getrokken worden, moeten wel onderzocht worden omdat:

- Een 100% dekking van de programmastatements garandeert geenszins dat er geen fouten meer kunnen voorkomen! Zie paragraaf 14.2.2 “Dekking, dekkingsvorm en dekkingsgraad”;
- Een test die ontworpen is om 100% dekking van de functionele specificaties te bereiken, zal in het algemeen niet automatisch ook 100% statement coverage bereiken.

Comparator

Een comparator wordt ook wel een vergelijkingstool genoemd. Deze vergelijkt data en rapporteren de verschillen. Deze verschillen moeten vervolgens handmatig worden geanalyseerd om te bepalen of het verschil overeenkomt met de verwachting. Deze tools worden bijvoorbeeld gebruikt om:

- de testuitvoer te vergelijken met de testuitvoer van de vorige test;
- een database vóór en ná één of meer testactie(s) te vergelijken;
- de resultaten van de schaduwproductie te vergelijken met de resultaten van productie.

De comparator is vaak een integraal onderdeel van record&playback tools. Als alternatief zijn de eenvoudige file-compare functionaliteit[4] of de revisiefunctionaliteit van een tekstverwerker te gebruiken.

Databasemanipulatietool

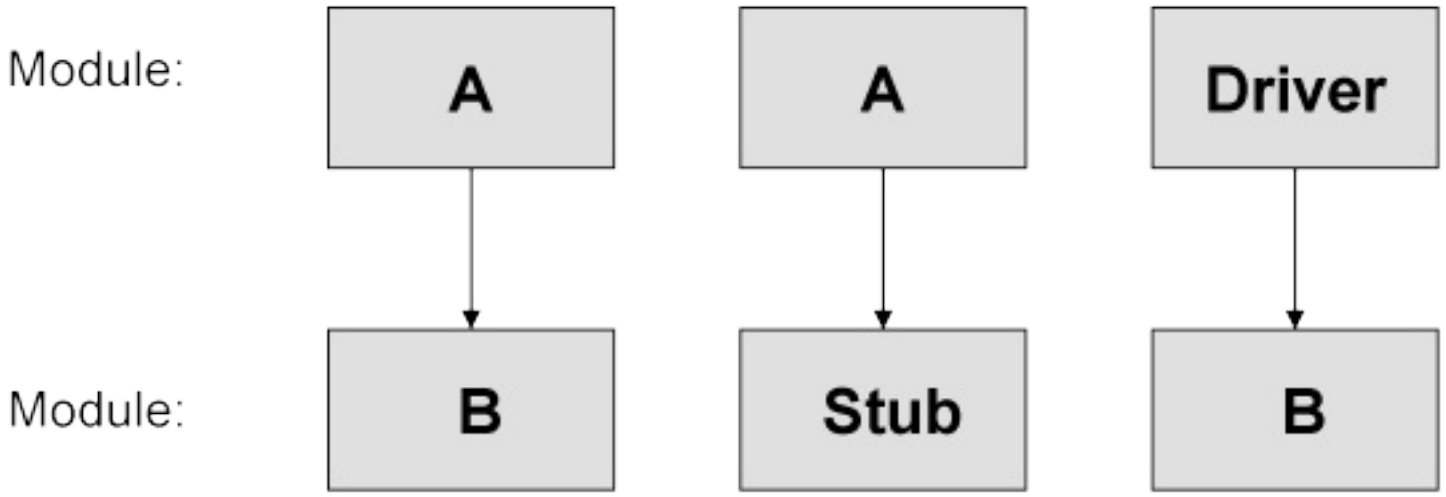
Het rechtstreeks bekijken en manipuleren van gegevens in een database is een krachtig instrument voor testers. Op deze manier kunnen controles worden uitgevoerd om te kijken of een test werkelijk succesvol is verlopen. Dit soort tools vormt een essentieel onderdeel van de standaarduitrusting van iedere tester. Naast het opvragen van gegevens kunnen deze ook worden gewijzigd. Dit kan gebruikt worden bij het maken van uitgangsituaties. De manipulatietaal waarop dit soort tools zijn gebaseerd is vaak SQL[5].

Simulator

Een simulator bootst de werking van de omgeving van het te testen (deel van het) testobject na. Een simulator wordt gebruikt om software te testen waarvan het te kostbaar, te gevaarlijk of zelfs onmogelijk is om deze in de werkelijke omgeving te testen, bijvoorbeeld het testen van de besturingssoftware van een vliegtuig of kernreactor. De simulator communiceert met het testobject alsof het de werkelijke omgeving is. Het levert invoer aan het testobject en vangt de uitvoer van het testobject weer op. Simulatoren zijn meestal niet standaard en moeten ontwikkeld worden parallel aan de ontwikkeling van het testobject. De simulator moet op zijn beurt ook weer getest worden.

Stubs en drivers

Een systeem wordt meestal getest in delen. Een deel kan bijvoorbeeld een module of component zijn. Om in een vroegtijdig stadium een module te kunnen testen die relaties heeft met nog niet gerealiseerde modules, zijn stubs en drivers nodig die de ontbrekende modules vervangen. Een stub wordt aangeroepen vanuit de te testen module, een driver roept de te testen module juist aan (zie figuur 8.16 “stubs en drivers in relatie tot module A en module B”).



Figuur 8.16: stubs en drivers in relatie tot module A en module B.

Voorbeeld

Een rapportagefunctie, die per medewerker het salaris afdrukt, wordt getest. Binnen deze functie wordt het (al eerder geteste) salarisberekeningsprogramma aangeroepen. Het gaat er bij de bedoelde test alleen om dat alle medewerkers worden geselecteerd en dat voor elke medewerker het salaris wordt afgedrukt. Het klaarzetten van een testdatabase met alle benodigde gegevens voor de diverse salarisberekeningen kan echter een enorm werk zijn. Een stub die een bepaald salarisbedrag (bijvoorbeeld gebaseerd op het ingegeven medewerkernummer) teruggeeft, kan de testinspanning aanzienlijk verlichten. Wél moet natuurlijk altijd de relatie tussen de echte programma's een keer getest worden.

8.5.4 Voordelen gebruik testtools

Met de invoering van een testtool wordt een geautomatiseerd hulpmiddel geïntroduceerd, dat ondersteuning biedt aan één of meer testactiviteiten. De invoering van testtools kan verschillende voordelen hebben en deze voordelen worden hier toegelicht. De besproken voordelen zijn niet van toepassingen op alle hiervoor besproken toolsoorten. Dit wordt bij het betreffende voordeel toegelicht.

Hogere productiviteit

Door het gebruik van een testtool voor routinematig testwerk heeft de tester veel meer tijd voor andere zaken. Zeker bij tools voor geautomatiseerde testuitvoering kan een omvangrijke hoeveelheid tests 'onbewaakt' en sneller worden uitgevoerd, bijvoorbeeld 's nachts. Dit betekent dat er veel grondiger op verschillende gebieden kan worden getest, maar ook dat de testomgeving efficiënter gebruikt kan worden. De testomgeving wordt dan 24 uur per dag gebruikt, waarbij 's nachts de testuitvoering plaatsvindt en overdag de resultaten van de testuitvoering onderzocht worden.

Tip

Nachtelijke back-ups

Het maken van een back-up gebeurt meestal in de nachtelijke uren. Dan worden de systemen niet gebruikt en zijn de meeste middelen, zoals CPU en geheugen, beschikbaar. Wanneer een geautomatiseerde test 's nachts uitgevoerd moet worden is het van belang om vooraf uit te zoeken of er in die periode back-up's worden gemaakt. Deze kunnen de geautomatiseerde testuitvoering namelijk blokkeren en dit wordt dan pas de volgende ochtend ontdekt. Dit geldt ook voor interfaces en andere systemen die 's nachts worden stopgezet.

Toolsoorten waar gegevens worden beheerd (bijvoorbeeld testwarebeheertools en bevindingenbeheertools) nemen de routinematige beheeractiviteiten over van de tester. Ook de arbeidsintensieve taak voor het creëren van rapportages wordt, met het gebruik van deze tools, teruggebracht naar één spreekwoordelijke druk op de knop. Hierdoor blijft meer

tijd over voor andere activiteiten.

Hogere kwaliteit testen

Het toepassen van testtools ter ondersteuning van een gestructureerde testaanpak is nadrukkelijk een volgende stap in de richting van hoge testkwaliteit en kwaliteitssoftware. De reden hiervoor is de consequente uitvoering van een activiteit die wordt ondersteund door de tool. Een tool dwingt tot een standaard manier van werken en hierdoor wordt de menselijke factor uitgeschakeld. Deze menselijke factor zorgt ervoor dat niet elke handeling exact hetzelfde zal zijn als de andere, waardoor de kans op het maken van fouten wordt vergroot. Dit wordt vooral versterkt bij langdurige repetitieve activiteiten zoals het schrijven van gelijksoortige fysieke testgevallen of het uitvoeren van de testscripts. Door deze standaard manier van werken wordt ook alle informatie (input en output) van een activiteit op een vaste wijze gepresenteerd. Dit zorgt ook voor het minimaliseren van interpretatiefouten door verschillende testers binnen een team.

Uitgediept

Zeer regelmatig is het afwijken van de voorgeschreven stappen of handelingen aanleiding tot het doen van bevindingen. Juist die menselijke (onberekenbare) factor is daarbij dan net weer van belang. De inzet van alleen tools voor geautomatiseerd testen is dan ook nooit de ultieme oplossing voor het verkrijgen van een hoge kwaliteit test.

Meer arbeidsvreugde

Het uitvoeren van routinematige activiteiten wordt als vervelend ervaren. Wanneer deze activiteiten geautomatiseerd worden leidt dit, naast een grotere betrouwbaarheid, tot een hogere arbeidsvreugde van het testteam. Uiteraard heeft dit ook weer een hogere productiviteit van het testteam tot gevolg. Bovendien wordt het werken met tools als prettig ervaren door het testteam. De redenen hiervoor zijn de professionele uitstraling die het heeft naar de rest van de organisatie en de nieuwe taken (realisatie en onderhoud van de tools) die ontstaan bij het gebruik van de testtools.

Uitbreiding testmogelijkheden

Sommige testen zijn niet handmatig volledig na te bootsen. Een voorbeeld is het uitvoeren van stresstesten. Hierbij blijkt de inzet van testtools nagenoeg onmisbaar. Het inzetten van een groot aantal “echte” gebruikers om een representatieve bezetting te verkrijgen lukt met veel inspanning nog wel één keer, maar meestal geen tweede keer. Bovendien kunnen deze tools fouten traceren die handmatig slechts moeilijk te constateren zijn. Bijvoorbeeld het meten van het geheugengebruik van een website wanneer er meer dan 1000 bezoekers zijn. Een ander voorbeeld is het testen van een systeem zonder gebruikersinterface (bijvoorbeeld middleware). Door het gebruik van een testtool kan dit systeem wel worden benaderd.

8.5.5 Invoeren van testtools met toolbeleid

Toolbeleid

Welke activiteiten van het testproces worden ondersteund door een testtool en hoe dit wordt ingericht, hangt af van het toolbeleid dat wordt gekozen binnen de organisatie.

Definitie

Het toolbeleid beschrijft hoe een organisatie omgaat met de aanschaf, de invoering en het gebruik van testtools in de verschillende situaties.

Het toolbeleid maakt deel uit van het overkoepelende testbeleid (zie paragraaf 8.2 “Testbeleid”). In het toolbeleid wordt op eenduidige wijze beschreven wat het doel moet zijn van de inzet van testtools in een organisatie. Het gebruik van testtools is nooit een doel op zich. De testtool is slechts een middel om een specifiek doel te realiseren in termen van tijd,

geld en/of kwaliteit. Dit doel wordt testtooldoel genoemd. Ook staat in het toolbeleid beschreven wat eventuele eisen, wensen en voorwaarden zijn, die worden gesteld aan testtools. Deze kunnen worden ingegeven door eisen, wensen en voorwaarden die worden gesteld in het testbeleid.

Daarnaast beschrijft het toolbeleid ook de te volgen aanpak bij de aanschaf, inzet en gebruik van tools. Dit deel van het toolbeleid lijkt daarmee op een algemeen plan van aanpak alleen met dat verschil, dat het de basis vormt voor een lange termijn investering. Het is al vaker geschreven, de inzet van tools verdient zich vaak alleen op lange termijn terug en daarom ligt er een beleid aan ten grondslag. Een toolbeleid vormt de basis waarop de organisatie (in de toekomst) het gebruik en de inzet van tools kan baseren. Het toolbeleid is geen eenmalig document dat in de kast verdwijnt. Het moet continu geactualiseerd en aangepast worden op nieuwe ontwikkelingen en inzichten.

Voorbeeld 1

Een pakketleverancier heeft als beleid om bouw- en testtools zoveel mogelijk van één leverancier te betrekken. Dit wordt verwerkt in het toolbeleid waarin een lijst wordt opgenomen van toolleveranciers die de voorkeur hebben.

Voorbeeld 2

Een organisatie heeft als doelstelling om binnen 3 jaar alle gebruikte systemen over te zetten naar een nieuwe hard- en softwareconfiguratie. Daarom wordt in het toolbeleid opgenomen dat nieuwe testtools alleen aangeschaft mogen worden indien deze ook werken op de nieuwe hard- en softwareconfiguratie. Ook wordt in het toolbeleid een bepaling opgenomen dat de inzet van geautomatiseerde testuitvoering binnen 2 jaar moet lonen. De reden hiervoor is dat systemen bij het overzetten naar de nieuwe hard- en softwareconfiguratie zullen wijzigen. De geautomatiseerde testuitvoering zal hierdoor ook moeten veranderen.

Voorbeeld 3

Een beursgenoteerde organisatie moet voldoen aan wettelijke bepalingen. Hierin staan voorwaarden vermeld, waaraan de systemen van de organisatie moeten voldoen. Het testen van deze voorwaarden kan alleen met bepaalde testtools die ook weer aan deze voorwaarden moeten voldoen. De lijst van testtools die hieraan voldoen wordt opgenomen in het toolbeleid.

Voorbeeld 4

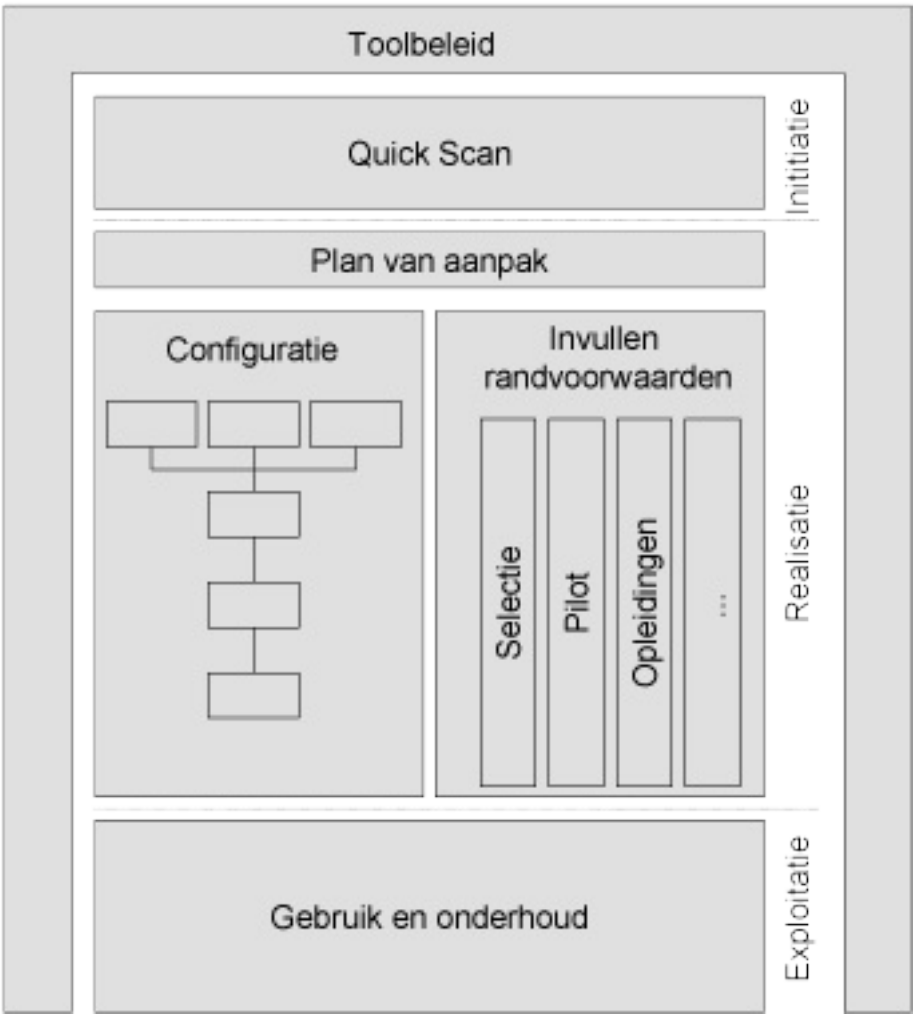
Een middelgrote organisatie heeft in zijn toolbeleid de bepaling opgenomen, dat er binnen de organisatie geen kennis hoeft te zijn rond de inzet van load-, performance- en stresstesttools. Dit resulteert in het feit dat alle performancetesten worden uitgevoerd door een externe leverancier.

Voorbeeld 5

Een energieleverancier heeft in zijn toolbeleid staan dat elk project gebruik moet maken van de standaard beschikbare testwarebeheertool. Andere tools moeten bij voorkeur open-source tools zijn. Alleen met toestemming van de manager IT mogen commerciële tools worden aangeschaft.

Faseringsmodel voor het invoeren van tools

Het invoeren van testtools in een gestructureerd testproces is een traject dat uit een aantal processtappen bestaat. Deze processtappen zijn ondergebracht in een faseringsmodel (zie figuur 8.17 “Faseringsmodel voor de invoering van testtools”). Hierin zijn de activiteiten gedefinieerd van de prille start van het gebruik van testtools tot en met de structurele inbedding in het testproces. In het faseringsmodel worden drie fasen onderkend: *Initiatie*, *Realisatie* en *Exploitatie*. Elke fase kent haar eigen doelstellingen, activiteiten en producten en deze worden getoetst aan het overkoepelende toolbeleid. De fasen worden in de volgende paragrafen toegelicht.



Figuur 8.17: Faseringsmodel voor de invoering van testtools.

8.5.6 Fase Initiatie

De eerste fase binnen het faseringsmodel is de fase Initiatie. Binnen deze fase worden activiteiten onderkend die tot doel hebben een eenduidig beeld te verkrijgen van de toepasbaarheid van een testtool in een specifieke situatie. Belangrijke voorwaarde voor de toepasbaarheid vormt de informatie die in het toolbeleid staat. Op basis hiervan wordt op verantwoorde wijze een beslissing genomen over de inzet van en investering in testtools. De belangrijkste activiteit van de fase Initiatie is de uitvoering van de *quick scan*. Deze quick scan geeft informatie over de technische omgeving, de volwassenheid van het testproces en de verwachtingen van het management ten aanzien van de inzet van testtools. Kenmerken van de quick scan zijn de beperkte doorlooptijd en de relatief beperkte investering. Naast het uitvoeren van de quick scan kunnen diverse andere activiteiten worden uitgevoerd. Hierbij kan worden gedacht aan productpresentaties, demo-sessie en ook het bezoeken van reeds werkende testtools bij andere organisaties.

De quick scan

De quick scan is het instrument dat wordt gehanteerd voor het verkrijgen van de specifieke informatie rond de inzet van testtools. Het is nog niet bepaald of er tools gebruikt gaan worden en welke tools dat moeten zijn. Doel van de quick scan is om met een beperkte inspanning (duur van 2 tot 15 dagen) informatie te verzamelen en te rapporteren, over de mogelijke toepasbaarheid van een testtool in een specifieke situatie. Dit resulteert in een eerste (ruwe) versie van de zogenaamde business case rond de inzet van tools.

Een belangrijke bron van informatie tijdens de quick scan vormen de interviews. Deze worden afgenomen met de belangrijkste belanghebbenden bij het testproces. Voorbeelden zijn:

- Lijnmanager (verantwoordelijk voor financiën);
- Projectmanager;
- Testmanager;
- Testconsultant;

- Applicatiedeskundige;
- Ontwikkelaar;
- Technisch Systeembeheer.

Naast het houden van interviews worden tijdens de quick scan ook diverse testproducten beoordeeld op bruikbaarheid in een testtool. Er wordt onderzocht in hoeverre bestaande producten, zoals bijvoorbeeld testgevallen of bevindingenprocedures, aansluiten op de manier van werken van testtools. Er wordt gelet op een drietal aspecten die van belang zijn voor de bepaling van toepasbaarheid van een testtool. Dit zijn:

- Testtooldoelen:
  In de quick scan wordt geïnventariseerd in welke mate de testtooldoelen reeds zijn gesteld en of ze aansluiten op de doelen zoals ze in het toolbeleid staan beschreven. Regelmatig is op dat moment nog slechts sprake van verwachtingen bij de betrokkenen, die of niet realistisch blijken te zijn of niet te vertalen zijn naar concrete doelen. Een randvoorwaarde voor het succesvol invoeren van testtools (zie ook paragraaf 8.5.7 “Fase Realisatie”) is een opdrachtgever die zich bewust is van de kansen in het huidige testproces. Deze kansen zijn de basis voor concrete verbeterdoelstellingen. Op basis van deze verbeterdoelstellingen worden de doelstellingen van de invoering van een testtool verder opgesteld en zoveel mogelijk geconcretiseerd. Het hier concretiseren van de doelen biedt later mogelijkheden tot het meetbaar maken van de behaalde resultaten.
- Infrastructuur en testobject:
  De infrastructuur (testomgeving, testwerkplek en eventueel andere tools) en het testobject spelen een cruciale rol in de bepaling van de toegevoegde waarde van een testtool. Zo moet er onderzocht worden of een tool wel aansluit op het testobject en de technische omgeving. Zeker wanneer er gekozen wordt voor geautomatiseerde testuitvoering is dit een vereiste. Belangrijk is ook om te onderzoeken of er speciale testtools voor het testobject bestaan en wat de mogelijkheden van deze producten zijn. Zeker in de situatie dat het testobject een standaardpakket is, komt dit nog al eens voor.
- Testaanpak:
  Inzet van tools (in het kader van efficiëntie) heeft vooral toegevoegde waarde als de processen voorspelbaar en herhaalbaar zijn. Bovendien moet de procesgang wel ondersteund worden door de tool. Een testtool die dient ter ondersteuning voor bevindingenbeheer moet bijvoorbeeld wel passen binnen het proces van bevindingenbeheer. Een onderdeel hierbij is het aspect van locatie. Wanneer een testorganisatie op verschillende fysieke locaties werkt, zal de tool hier ook ondersteuning voor moeten bieden. In het voorbeeld van de tool voor bevindingenbeheer moet deze toegankelijk zijn vanuit de verschillende locaties en alle testers moeten met dezelfde database werken.

Met de resultaten van de interviews en de beoordelingen van de diverse testproducten, kan een eerste (ruwe) versie van de business case worden opgesteld. Belangrijkste aspecten hierbij zijn de verwachte investeringen en de verwachte opbrengsten. Dit is een mix van tastbare en ontastbare zaken en daarom is het altijd zeer lastig een business case te maken. Bovendien zijn veel zaken na de quick scan nog niet bekend. Er is immers nog niet gekozen voor een bepaalde testtool. Een eerste versie van de business case zal daarom bestaan uit de baten die de belanghebbenden verwachten en een inschatting van de kosten die gemaakt moeten worden om de testproducten te gebruiken in een testtool. Ook kan een business case bestaan uit verschillende scenario’s waarin de inzet van verschillende toolsoorten wordt uitgewerkt.

Uitgediept

Tastbare en ontastbare opbrengsten van de inzet van tools

Het definiëren van de business case rond de inzet van testtools is altijd moeilijk. Deels omdat het gaat om vaste en variabele kosten, deels omdat het gaat om tastbare en ontastbare opbrengsten. Voorbeeld van een tastbare opbrengst is de verkorting van de doorlooptijd. Maar er zijn ook vaak indirecte baten die niet rechtstreeks met geld te maken hebben. Zo zal het imago van de testorganisatie verbeteren. Er wordt professionaliteit uitgestraald. Gebruikers en beheerorganisaties willen graag demonstraties zien van het geautomatiseerd testen. Er wordt vaker bij de testorganisatie aangeklopt om te helpen bij uiteenlopende gebeurtenissen. Medewerkers zullen meer gemotiveerd worden. Nieuwe carrièremogelijkheden ontstaan: technische specialisatie en werken met moderne tools.

Op basis van de resultaten van de interviews, de beoordeling van de diverse testproducten en de business case, wordt een rapport opgesteld. Dit rapport bevat, naast de business case, een conclusie gericht op eventuele inzet van testtools. Naast deze conclusie worden concrete aanbevelingen gedaan voor het vervolgtraject en welke stappen er door welke personen moeten worden genomen.

8.5.7 Fase Realisatie

De tweede fase binnen het model is de fase Realisatie. Op basis van een op te stellen *plan van aanpak* worden alle activiteiten uitgevoerd en producten gerealiseerd die nodig zijn om een testtool in een organisatie te gebruiken. Doel van de fase Realisatie is het implementeren van een testtool inclusief de benodigde *configuratie*. Een ander onderdeel van deze fase is het invullen van de *randvoorwaarden* om het gebruik mogelijk te maken. Voor de uitvoering van deze drie onderdelen zijn drie deelfasen onderkend en deze worden hier toegelicht. De drie deelfasen zijn:

1. Plan van aanpak;
2. Invullen randvoorwaarden;
3. Configuratie testtool.

Deze deelfasen worden parallel uitgevoerd. Zo kunnen bevindingen (door bijvoorbeeld voortschrijdend inzicht) uit een deelfase worden meegenomen bij de uitvoering van een andere deelfase. Deze manier van werken is tijdsbesparend.

Deelfase Plan van aanpak

De quick scan levert informatie om een eerste versie van het plan van aanpak op te stellen. Hierin staat beschreven hoe een eerste invulling wordt gegeven aan de randvoorwaarden. Aangeraden wordt om te starten met het uitvoeren van een testtoolselectie en het uitvoeren van de pilot. Deze worden expliciet opgenomen als activiteiten in het plan. Na afloop van de pilot kan het plan van aanpak worden geactualiseerd en geconcretiseerd. Belangrijkste doel van een plan van aanpak is het eenduidig definiëren van zaken als doelstelling, activiteiten, planning en op te leveren producten. Voorbeelden van onderwerpen die in het plan van aanpak worden benoemd zijn:

- Testtooldoel;
- Randvoorwaarden;
- Aanpak pilot;
- Aanpak configuratie;
- Activiteiten;
- Planning;
- Producten;
- Organisatie.

Deelfase Invullen randvoorwaarden

Om het gebruik van testtools mogelijk te maken moet aan een aantal randvoorwaarden voldaan zijn. De belangrijkste randvoorwaarde is natuurlijk het aanwezig zijn van een gestructureerd testproces, waarbinnen het gebruik van tools tot verbeteringen kan leiden. Naast randvoorwaarden die de inzet van de testtool mogelijk maken, zijn er randvoorwaarden die ingevuld moeten zijn om het gebruik van de tool door de testers mogelijk te maken. Niet alleen voor de huidige testactiviteiten moeten de testers in staat zijn de tool te gebruiken, maar ook voor toekomstige testactiviteiten. De manier waarop invulling wordt gegeven aan de randvoorwaarden kan afhankelijk zijn van wat in het toolbeleid staat beschreven. Aan welke randvoorwaarden moet worden voldaan en hoe deze ingevuld moeten worden is situatiespecifiek. Toch zijn er een aantal algemeen geldende voorwaarden te identificeren:

- Testtoolselectie;
- Pilot;
- Business case;
- Management commitment;
- Onderhoud en beheer in de lijn;
- Opgeleide testers;
- Gestructureerd testproces;
- Communicatie.

Deze voorwaarden worden hierna verder uitgediept.

Testtoolselectie

Er is een groot scala van testtools die ingezet kunnen worden in een testproces. De specifieke omgeving en doelstellingen bepalen welke testtool(s) in die situatie het meest geschikt zijn. De (toekomst)strategie van de testtoolleverancier wordt ook steeds belangrijker bij de selectie van een tool. Doordat steeds meer testtools worden geïntegreerd, houdt de keuze van een product veelal ook de keuze voor een leverancier in. Als nog geen testtool beschikbaar is binnen het testtraject wordt een testtoolselectie uitgevoerd. Hiervoor zijn diverse aanpakken beschikbaar die sterke overeenkomsten vertonen met een reguliere pakketselectie. Op basis van een vooraf op te stellen lijst van criteria worden verschillende tools geëvalueerd. Welke criteria dit zijn hangt af van de toolsoort waarvoor de testtoolselectie plaats vindt. Een lijst met voorbeeldcriteria staat op www.tmap.net onder de naam "selectiecriteria testtools".

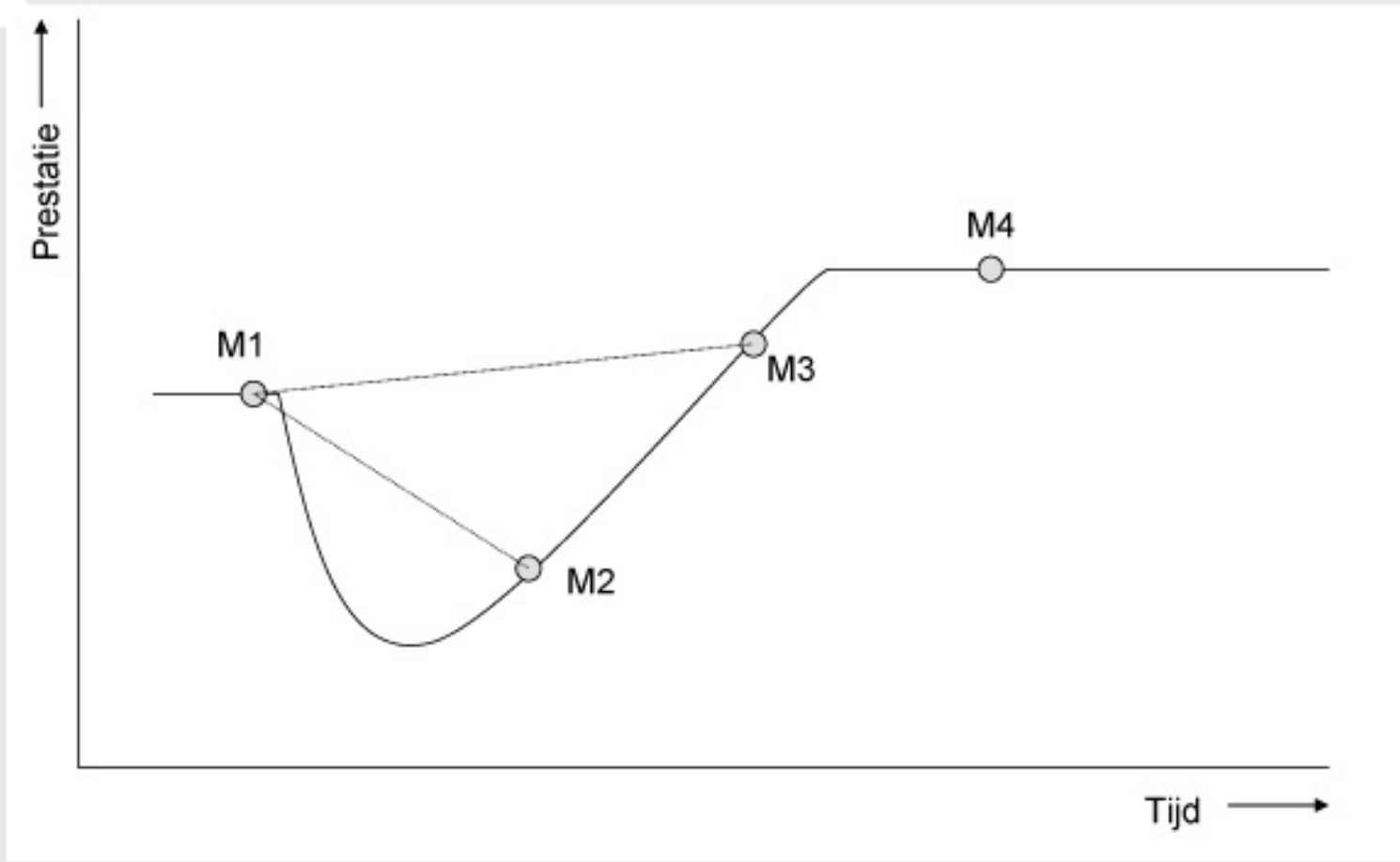
Pilot

Het invoeren van een testtool is geen standaard proces dat in iedere situatie op dezelfde wijze kan plaatsvinden. Elk traject kent zijn eigen valkuilen. Vaak zijn de verwachtingen van testtools bij veel mensen zeer hoog. Men is zich veelal niet bewust dat de inzet van tools een investering vereist, waarvan de baten veelal niet op korte termijn zichtbaar zijn. Bij de inzet van tools moet daarom zeer zorgvuldig te werk gegaan worden, om niet ten onder te gaan aan het verschil tussen verwachtingen en realiteit. Door te starten met een pilotproject wordt in een relatief beperkte omgeving op korte termijn inzicht gegeven in de toegevoegde waarde van een testtool. Binnen een pilot kan op kleine schaal de tool worden gebruikt. Bijvoorbeeld door een deel van het team of het testen van een specifieke functie. Zo kan de haalbaarheid van de testtooldoelen worden getoetst. Met een beperkte inspanning wordt inzichtelijk gemaakt of de testtool technisch haalbaar is, of het aansluit op de huidige wijze van testen en wat de te verwachten kosten en baten zijn.

Uitgediept

De dip in de prestatiecurve of waarom een pilot nodig is

Wat zijn de gevolgen van de invoering van testtools voor de medewerkers? Dat kan duidelijk gemaakt worden met figuur 8.18 "Prestatiedip bij de introductie nieuwe werkwijze". De figuur beschrijft de situatie waarin een organisatie beter wil gaan presteren op een bepaald gebied. Op dit moment wordt gepresteerd op niveau M1. De wens van de organisatie is om te presteren op niveau M4. Om dit te bereiken wordt een nieuwe werkwijze geïntroduceerd.



Figuur 8.18: Prestatiedip bij de introductie nieuwe werkwijze.

In figuur 8.18 is te zien hoe de introductie van een nieuwe werkwijze in eerste instantie een dip in de prestatiecurve veroorzaakt. De weg om te komen van presteren op het niveau M1 naar presteren op niveau M4 loopt niet in een rechte lijn omhoog. Een nieuwe werkwijze moet eerst nog aangeleerd worden en in de meeste gevallen ook nog worden aangepast aan de specifieke situatie. Wanneer de betrokkenen zich er niet van bewust zijn dat het prestatieniveau in eerste instantie zal dalen, dan bestaat het gevaar dat te vroeg wordt gemeten. De (lagere) baten van niveau 2 worden dan gemeten. Er wordt dan geconcludeerd dat dit niet de juiste werkwijze is en dat een andere oplossing moet worden gekozen. Bij testtools wordt dan vaak het gebruik stopgezet en verdwijnt de tool in de kast (ook wel bekend als “shelfware”).

Het is aan te bevelen om te gaan meten wat de baten zijn van de nieuwe werkwijze, wanneer de stijgende lijn, ten opzichte van het niveau M1, is ingezet. In dit voorbeeld is dat punt M3. Dit is een inschatting die lastig te maken is. Wanneer bij de introductie van een nieuwe werkwijze dan ook nog eens gekozen wordt voor een lang veranderingstraject, dan is het gevaar van te vroeg meten en het trekken van de verkeerde conclusies nog groter. Dat is één van de redenen om hiervoor een kortlopende pilot te gebruiken. Tijdens de pilot zal de dip er zeker zijn, maar op basis van de leerpunten en bevindingen van de pilot zal de dip in een “echte productie situatie” kleiner (in ieder geval beter voorspelbaar) zijn.

Business case

De eerste versie van de business case, die is opgesteld in de fase Initiatie, wordt verder ingevuld. De cijfers binnen de business case kunnen nu concreet zijn. Bij het benoemen van de kosten moet rekening gehouden worden met de vaste kosten en de variabele kosten. Vaste kosten kunnen zijn: hardware, licenties, installatie, onderhoud en opleiding. Variabele kosten kunnen zijn: creëren testscripts, uitvoeren testscripts, analyse resultaten, beheren testscripts en opleidingen.

Management commitment

Zelfs als het testen voldoende volwassen is voor het toepassen van tools, is het niet altijd even zeker dat de gewenste voordelen worden behaald. Eén van de belangrijkste succesfactoren bij de inzet van testtools is het commitment van het management. Het management dient bewust te worden gemaakt dat het gebruik van de tool een investering is, die vaak pas op langere termijn wordt terugverdiend in termen van sneller en/of beter testen. Bij onvoldoende bewustzijn is het risico groot dat verder gebruik van de tool bij de eerste de beste tegenvaller wordt stopgezet. Dit speelt in versterkte mate wanneer men de tool voor het eerst inzet in een project met vaste einddatum. Wanneer het project dan in tijdnood komt, is de kans groot dat het gebruik van de tool wordt stopgezet.

Implementeren van onderhoud en beheer in de lijn

Een operationele testtool kan uit een groot aantal items bestaan: modules in de testtool, testgegevens, documenten voor gebruik en beheer enzovoort. Voor al deze items moet beheer worden uitgevoerd om hergebruik in de toekomst mogelijk te maken. Het inzetten van testtools loont alleen over langere perioden en dus vaak over verschillende projecten heen. Door het beheer en onderhoud van tools te beleggen in de lijn wordt kennisbehoud gegarandeerd.

Opgeleide testers

Wanneer het werken met een tool nieuw is binnen het testteam moeten de testers worden opgeleid. Zowel de tool als het werken met een tool zijn nieuwe begrippen voor het testteam. Om het gebruik en onderhoud goed te laten verlopen is het noodzakelijk dat de testers kennis opdoen. Het opleiden van de medewerkers richt zich dan ook op een tweetal zaken: kennis opdoen van de tool en kennis opdoen rond het gebruik van de tool in het testproces.

Gestructureerd testproces

De inzet van testtools richt zich op het verbeteren van het testproces in termen van geld, tijd en kwaliteit. Het levert daarmee een bijdrage aan de verhoging van de efficiëntie van het testproces. Alvorens een testproces efficiënter kan worden ingericht, moet het beheerst worden uitgevoerd. Het kan noodzakelijk zijn om aanvullende activiteiten te definiëren in het kader van de beheersing van het testproces. Hierbij kan bijvoorbeeld worden gedacht aan het gebruik van testontwerptechnieken (zie hoofdstuk 14 “Testontwerptechnieken”) waardoor ook de traceerbaarheid van het testproces wordt vergroot. De uit te voeren maatregelen zijn zeer situatie specifiek. Het verdient aanbeveling om bij het

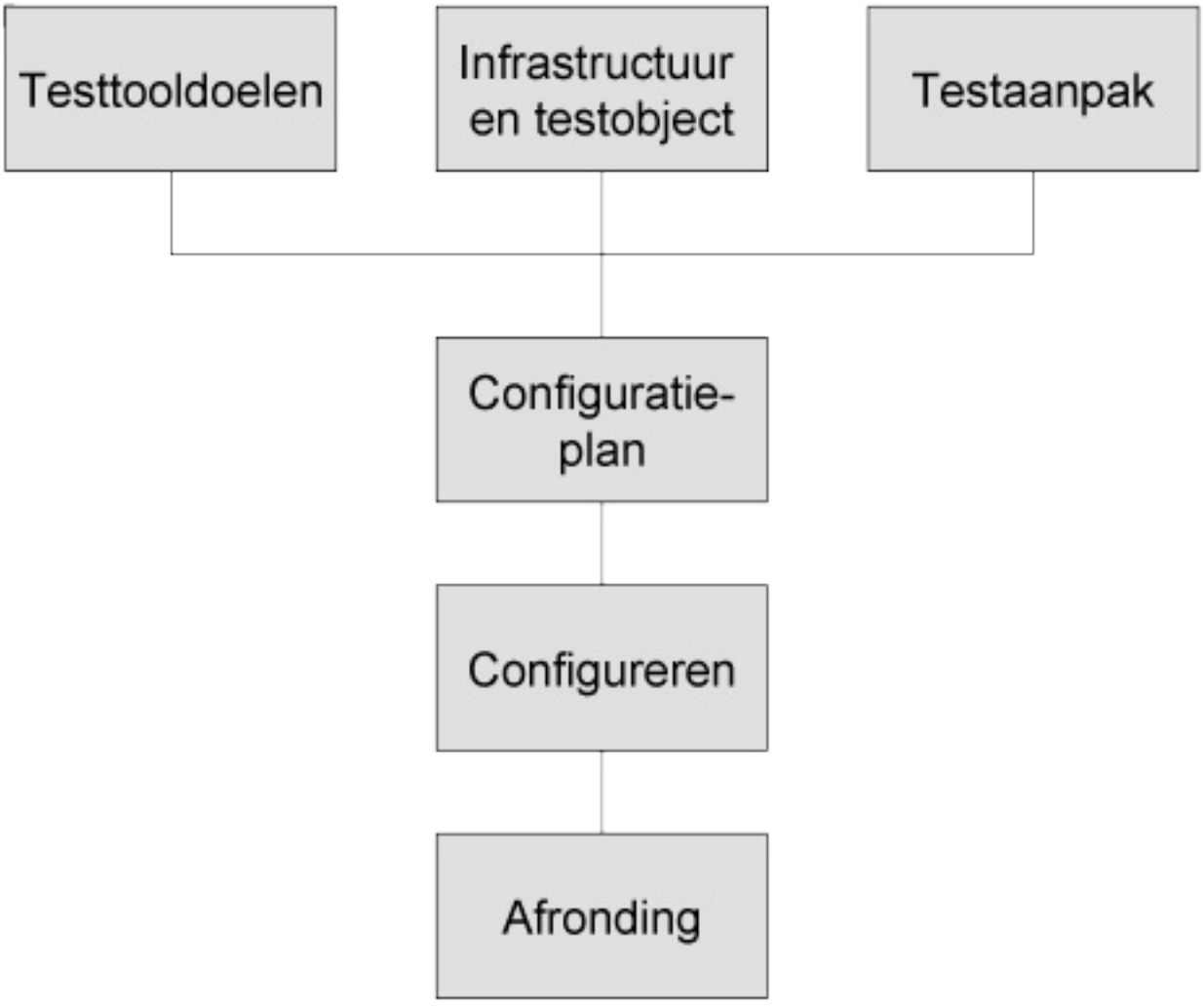
doorvoeren van de verbetering gebruik te maken van een verbetermodel (bijvoorbeeld TPI).

Communicatie

Wanneer binnen het testteam en de rest van de organisatie onbekendheid bestaat rond het werken met testtools dan verdient het aspect communicatie bijzondere aandacht. Zo vroeg mogelijk worden de betrokkenen voorgelicht over de plannen op het gebied van testtooling. Wat zijn de plannen, waarom worden ze uitgevoerd, wie voert ze uit, wat zijn de geplande resultaten en wanneer worden ze bereikt. Het heeft sterk de voorkeur om voor deze communicatie gebruik te maken van de reeds aanwezige communicatiemiddelen. Hierbij valt te denken aan: een regulier werkoverleg, een nieuwsbrief en het intranet. Als deze mogelijkheden niet aanwezig zijn, kunnen eigen informatiesessies georganiseerd.

Deelfase Configuratie testtool

In veel gevallen ondersteunen testtools een eigen manier van werken. Vaak wijkt deze manier van werken af van de situatie binnen de organisatie die deze tool gaat gebruiken. Daarom moet de tool geconfigureerd worden (zie figuur 8.19 “Configuratie van de testtool”). Het configureren van een testtool, zodat het aansluit op de organisatie, is maatwerk. Het omvat bijvoorbeeld activiteiten als het instellen van standaardtabellen, het definiëren van een workflow of het programmeren van een testsuite voor geautomatiseerde testuitvoering. De basis voor de configuratie wordt gevormd door de drie aspecten die in de voorgaande fase Initiatie zijn onderzocht. Dit zijn de aspecten testtooldoelen, infrastructuur en testobject en testaanpak. Op basis hiervan wordt een configuratieplan gemaakt. Hierin staat concreet beschreven hoe de tool geconfigureerd wordt. Dit is essentieel om de tool en de specifieke configuratie in de toekomst te onderhouden. Hierna wordt de tool op basis van het plan ge configureerd. Het is aan te bevelen dit te laten doen door of in samenwerking met de toekomstige beheerder van de tool. Op deze wijze vindt al de eerste kennisoverdracht plaats. Tijdens de configuratie van de tool moet de configuratie ook getest worden. De eventuele bevindingen uit deze test worden dan opgelost of tijdens de afronding verwerkt in het configuratieplan als bekende problemen. Bij een nieuwe versie van de tool kan dan onderzocht worden of deze problemen alsnog opgelost kunnen worden.



Figuur 8.19: Configuratie van de testtool.

8.5.8 Fase Exploitatie

De derde fase binnen het model is de fase Exploitatie. Deze fase start op het moment dat de testtool in gebruik genomen wordt door het testteam. Om ervoor te zorgen dat de testtool gebruikt kan blijven worden, moet er beheer worden uitgevoerd. Het gebruik van de testtool gaat onderdeel uitmaken van het reguliere testproces. Dit betekent dat er nieuwe activiteiten uitgevoerd moeten worden door zowel de testers als de testmanager. Dit betekent ook dat deze mensen in staat moeten zijn om de testtool op de juiste wijze te gebruiken en te beheren. De tool moet een plaats krijgen in het reguliere testproces. Bij het gebruiken van de testtool moeten gegevens verzameld worden omtrent de werking van de tool. Sluit de functionaliteit aan bij de manier van werken in het grotere geheel? Is dit niet het geval dan moet onderzocht worden of dit alsnog veranderd kan worden. Dit geldt ook voor de evaluatie van de testtooldoelen die zijn gesteld in de fase Initiatie. Op regelmatige tijden moet onderzocht worden of deze doelen nog gehaald worden met de inzet van de tool.

Eén van de belangrijkste uitgangspunten bij het gebruik van testtools is het aspect van de onderhoudbaarheid. In de fase Exploitatie vindt het daadwerkelijke onderhoud plaats. Wanneer gekozen is voor geautomatiseerde uitvoering zullen zaken als nieuwe releases, wijzigingen en incidenten van het testobject, impact hebben op de ingerichte testsuite. Maar ook nieuwe releases van de testtool zelf kunnen wijzigingen tot gevolg hebben. Deze wijzigingen kunnen eventueel worden doorgevoerd, waarna de testtool weer gereed is voor gebruik.

Er zijn drie vormen van beheer te onderscheiden:

- Technisch beheer
  Het installeren van de testtool (op de server of op werkplekken), implementatie van nieuwe versies of patches, oplossen technische incidenten enzovoort. De verantwoordelijkheid hiervoor ligt vaak bij de beheerafdeling die ook technisch beheer voert van andere applicaties binnen een organisatie.
- Operationeel beheer
  Het mogelijk maken dat gebruikers met de testtool kunnen werken. Dit kan zijn het verzorgen van autorisaties of het instellen van projectspecifieke onderdelen (bijvoorbeeld database). De verantwoordelijkheid hiervoor kan liggen bij de beheerafdeling die ook technisch beheer doet van de tool. Een andere mogelijkheid is om dit binnen het testproject zelf te beleggen of bij een permanente testorganisatie.
- Functioneel beheer
  De mogelijkheid dat gebruikers ‘goed’ met de tool kunnen werken. Dus het opstellen van werkinstructies, handleidingen voor de ‘eigen’ manier van werken, procedures, templates enzovoort. Belangrijk is dat functioneel beheer niet de functionaliteit van de tool zelf beheert, dat ligt bij de leverancier. De verantwoordelijkheid hiervoor kan liggen binnen het testproject of bij een permanente testorganisatie.

Deze drie beheervormen kennen een scheiding van verantwoordelijkheden, maar er moet uiteraard ook samengewerkt worden. Bijvoorbeeld bij een nieuwe versie van de testtool beoordeelt functioneel beheer de toegevoegde waarde en de impact. Aan de hand daarvan bepaalt functioneel beheer of deze nieuwe versie ingevoerd moet worden en wanneer dat plaats moet vinden. Functioneel beheer stuurt hierbij technisch beheer aan om de implementatie te verzorgen.

8.6 Testprofessionals

8.6.1 Inleiding

Om als tester goed invulling te kunnen geven aan het testvak is een grote verscheidenheid aan expertise nodig. Zo moet een tester kennis hebben op het gebied van:

- de materie (bijvoorbeeld logistieke processen en financiële rapportages);
- de infrastructuur (testomgeving, ontwikkelplatform, testtools);
- het testen zelf.

Het is aan het management om de juiste persoon met de juiste expertise op de juiste functie te krijgen, bij voorkeur in goede samenwerking met personeels- en opleidingsdeskundigen. Een zorgvuldige instroom en doorstroom, ondersteund met bijbehorende opleidingen van het testpersoneel, is vereist. Echter door het slechte imago van testen is geschikt en ervaren testpersoneel schaars.

Door deze combinatie van enerzijds het negatieve beeld en anderzijds het belang van testen ontstaat de uitdaging voor HRM; wie te vinden om deze taak uit te voeren en vooral ook, hoe deze tevreden te houden. Een belangrijk hulpmiddel voor deze tevredenheid is het bieden van een carrière aan de tester.

Dit hoofdstuk gaat in op hoe invulling te geven aan dit vraagstuk. In paragraaf 8.6.2 “Aandachtspunten” worden een aantal punten besproken die aandacht verdienen bij de inrichting van HRM voor testprofessionals. Paragraaf 8.6.3 “Eigenschappen” beschrijft wat een tester nou een tester maakt. Wat zijn bijvoorbeeld de persoonlijke kenmerken van een tester? De volgende paragraaf (8.6.4 “Carrièrepad”) geeft inzicht in een mogelijke carrière structuur voor een tester met daaropvolgend de paragraaf (8.6.5 “Functies”) met mogelijke functies. Als laatste (paragraaf 8.6.6 “Opleidingen”) wordt het aspect van opleidingen besproken.

8.6.2 Aandachtspunten

Ondanks het feit dat tegenwoordig iedereen wel het nut en de toegevoegde waarde van testen inziet, is het imago ervan niet in alle organisaties goed. Soms wordt een testfunctie gezien als saai, geestdodend en weinig uitdagend.

Of het wordt gezien als een eindpunt van je carrière of een noodzakelijk uitstap als er echt niks anders meer is. Hier volgen een aantal aandachtspunten voor de inrichting van HRM bij testprofessionals.

Taken, bevoegdheden en verantwoordelijkheden

Bij veel organisaties bestaat de situatie dat voor rollen en functies een uitgebreid competentieprofiel is geschreven, welke groeimogelijkheden (in rol en salaris) er mogelijk zijn en welke cursussen gevolgd kunnen worden. Voor de testers ontbreekt dat soms met bijvoorbeeld het excuus dat testen een eenmalige activiteit is. Dat is natuurlijk niet het geval. Ook voor testers is er een beschreven carrièrestructuur nodig.

Uitgediept

Doorgroeimogelijkheden van een tester

Bij de functiebeschrijvingen voor testers moet duidelijk zijn welke doorgroeimogelijkheden er zijn zowel binnen als buiten het vakgebied van testen. Omdat testen op het kruispunt van vele professies acteert zijn er vele (externe) groeirichtingen mogelijk. Zo kan een tester, die regelmatig betrokken is bij het testen van een specifieke bedrijfstoepassing, uitgroeien naar procesanalist voor die specifieke materie. Iets wat ook vaak voorkomt is dat een ervaren testmanager wordt gevraagd projectmanager te worden.

Opleidingsmogelijkheden

Doordat testen een risicobeperkende maatregel is, is een tester daarmee op zichzelf ook een risico. Als de testprofessional niet juist of niet voldoende test kunnen bepaalde risico’s niet ondervangen worden. Daarom is het belangrijk dat een tester niet alleen weet wat goed gestructureerd testen is maar ook weet wat hij test. Kijkend naar de definitie van risico’s (zie hoofdstuk 9 “Productrisicoanalyse”) betekent dit dat de tester kennis moet hebben van de materie (schade) en technologie (faalkans) die ten grondslag liggen aan het systeem. Hij moet deze eigen maken en weten waar in het algemeen de risico’s liggen (bijvoorbeeld die ene berekening of die gekozen combinatie van architectuur en hardware). Concreet houdt dit in dat testers ook naar cursussen kunnen (moeten) gaan die betrekking hebben op bijvoorbeeld de tool waarin geprogrammeerd wordt of de bedrijfsprocessen waarvoor de oplossing wordt gebouwd.

Locatie werkplek

Doordat testen zich begeeft op het kruispunt van vele professies is veel contact met de professionals in deze vakgebieden. Door de testers in het midden te zetten worden twee vliegen in één klap geslagen. Ze krijgen de uitstraling dat ze werkelijk op het kruispunt zitten en op dat kruispunt hebben ze veel contact. Er zijn voorbeelden van een beter testproces door de fysieke verhuizing van het testteam naar het ‘midden’ van de organisatie. Hierdoor werd onder andere het wederzijds respect tussen de programmeurs en testers vergroot en dat kwam de kwaliteit ten goede.

Beoordelen

Beoordelingsgesprekken worden over het algemeen uitgevoerd door een meerdere met ervaring in de uitgevoerde taken van de beoordeelde. Alleen zo kan er een objectief beeld ontstaan van de afgelopen periode en kunnen er afspraken worden gemaakt. Bij veel disciplines in de IT wordt het ook op deze manier uitgevoerd. Een programmeur wordt beoordeeld door bijvoorbeeld een projectleider die vroeger zelf geprogrammeerd heeft. Of een informatieanalist wordt beoordeeld door een businessanalist met een gemeenschappelijk verleden. Testers worden regelmatig beoordeeld door de projectleider. Vanuit zijn huidige rol heeft hij er veel mee te maken maar zelf heeft hij nooit getest. Om alle denkbare schijn van tegenstrijdige belangen te voorkomen is het noodzakelijk een tester door een meerdere of direct betrokkene te laten beoordelen met daadwerkelijke testervaring. Bijvoorbeeld de testcoordinator of testmanager.

Beloning

Een hedendaagse trend is om naast een vast salaris een variabel salaris in te stellen voor medewerkers. De hoogte van dit variabele deel (bonus) wordt afhankelijk gesteld van het halen van bepaalde resultaten (Key Performance Indicators of KPI's). Nu heeft een tester in een organisatie andere belangen dan bijvoorbeeld een informatieanalist, programmeur of projectleider. Een testprofessional moet op andere resultaten afgerekend worden. De situatie waarbij iedereen (inclusief tester) wordt afgerekend op het halen van de projectplanning is niet ideaal. De tester zit aan het einde en wordt vaak, ten onrechte, gezien als veroorzaker van de uitloop. Het is beter om een tester te beoordelen op het resultaat van zijn werk. Voorbeelden hiervan zijn het halen van zijn planning van één testronde of het aantal incidenten in productie. Natuurlijk gelden hier om heen een set aan uitgangspunten. Beloon een tester nooit op het aantal bevindingen die worden gedaan omdat dit weer afhankelijk is van de kwaliteit van de software (en dus weer iemand anders zijn aandachtspunt).

8.6.3 Eigenschappen

Wat is kenmerkend aan een tester of anders geformuleerd welke eigenschappen moet een persoon hebben om deze de ideale tester te kunnen zijn? Vooropgesteld, de ideale tester bestaat niet. Dat verschilt per situatie. Wel zijn er een aantal generieke eigenschappen te noemen:

Communicatief, mondeling en schriftelijk

De tester onderhoudt contacten met veel verschillende partijen. Zo zit hij om de tafel met bijvoorbeeld de programmeur, de informatieanalist, de projectleider en andere testers. Het is belangrijk dat de tester zich kan inleven in de belangen van de gesprekspartners en zich op de meest effectieve manier mondeling weet uit te drukken. Schriftelijke communicatie is van belang bij het vastleggen van bevindingen en het schrijven van rapporten.

Nauwkeurig en analytisch

Een tester moet oog hebben voor details. Belangrijk is om van elke requirement (eis) of wens nauwkeurig vast te stellen wat nu werkelijk gevraagd wordt en indien dit niet duidelijk is dit na te vragen. Het is belangrijk dat de tester analytisch te werk gaat en oppast met het doen van aannames. De basis voor zijn test wordt gevormd door de testbasis en wanneer deze niet compleet is of fouten bevat is dat een bevinding. Een tester mag hier nooit en te nimmer zelf aannames over doen, al liggen die soms voor de hand.

> **Voorbeeld**
>
> Gedurende een test van een financiële applicatie was er een requirement dat omschreef dat bedragen in euro's en dollars moesten worden weergegeven. De requirement omvatte een lijst met schermen waarop dat plaats moest vinden. Na een nauwkeurig analyse van de tester bleek dat er meer schermen waren waarop dit plaats kon vinden. Bij navraag bij de klant bleek dat de requirement inderdaad niet volledig was en daarop werd de lijst met schermen aangepast. Als de tester geen volledige analyse had gemaakt waren er onvolledige schermen in productie gegaan.

Overtuigend en volhardend

Een tester communiceert zijn bevinding aan een partij die deze bevinding heeft veroorzaakt. Hier speelt de manier van overtuigen een rol want de andere partij moet deze ook werkelijk als bevindingen zien. De tester moet durven staan voor zijn standpunten en volhardend zijn in het belang van de kwaliteit van het product.

Objectief en positief kritisch

Wanneer een bevinding wordt gecommuniceerd of vragen worden gesteld over een requirement is het belangrijk dit objectief te doen. Opmerkingen als ‘slechte software’, ‘weer al een foute requirement’ of ‘irritante kleuren’ mogen niet gebruikt worden. Bij discussies over bevindingen is het van belang dat de tester op een constructieve positieve manier de andere partijen duidelijk maakt waar het probleem zit. Dat betekent dus een bepaalde mate van tact en het voorkomen dat er met vingers naar verschillende partijen gewezen wordt.

Creatief

De tester moet de realiteit simuleren om zo een uitspraak te doen over de kwaliteit van de software. Voor deze simulatie worden testgevallen en testgegevens bedacht en een testomgeving gedefinieerd. Hiervoor is creativiteit vereist.

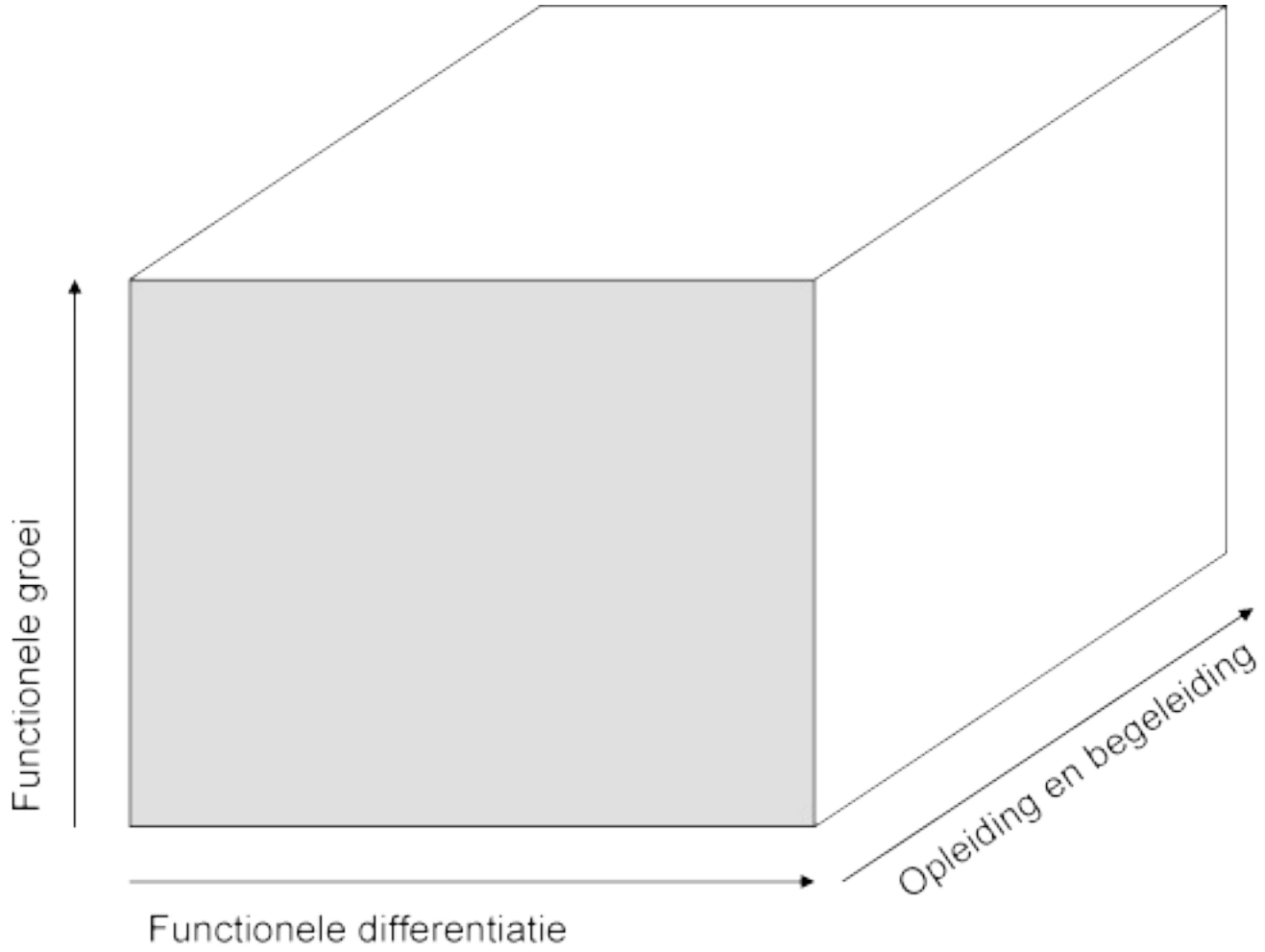
Sensitief

De tester zit op een kruispunt tussen professies. Het zwaartepunt van de werkzaamheden van de tester liggen aan het einde van een traject als de druk het grootst is. De tester dient zich bewust te zijn van de spanningen en de belangen om daar op de juiste manier mee om te gaan zodat de gewenste doelen bereikt kunnen worden.

8.6.4 Carrièrepad

Een voorwaarde om het testen in een organisatie te professionaliseren is het bieden van een carrièrepad aan testprofessionals. Deze paragraaf geeft een methode om deze carrièrepaden te benoemen. Hierbij zijn opleidingen, opbouw van werkervaring en een begeleidingsprogramma essentiële onderdelen. Deze worden per functie en per niveau aangegeven in de zogenaamde carrièrekubus (zie figuur 8.20 “De carrièrekubus”). De kubus is een hulpmiddel ten behoeve van carrièrebegeleiding door HRM om de vraag vanuit de organisatie, de beschikbare kennis en vaardigheden én de ambitie van de professionele testprofessionals op elkaar afgestemd te houden. De drie dimensies van de carrièrekubus worden als volgt gedefinieerd:

- Hoogte: functionele groei;
- Breedte: functionele differentiatie;
- Diepte: opleiding en begeleiding.



Figuur 8.20: De carrièrekubus.

Eerste dimensie: functionele groei

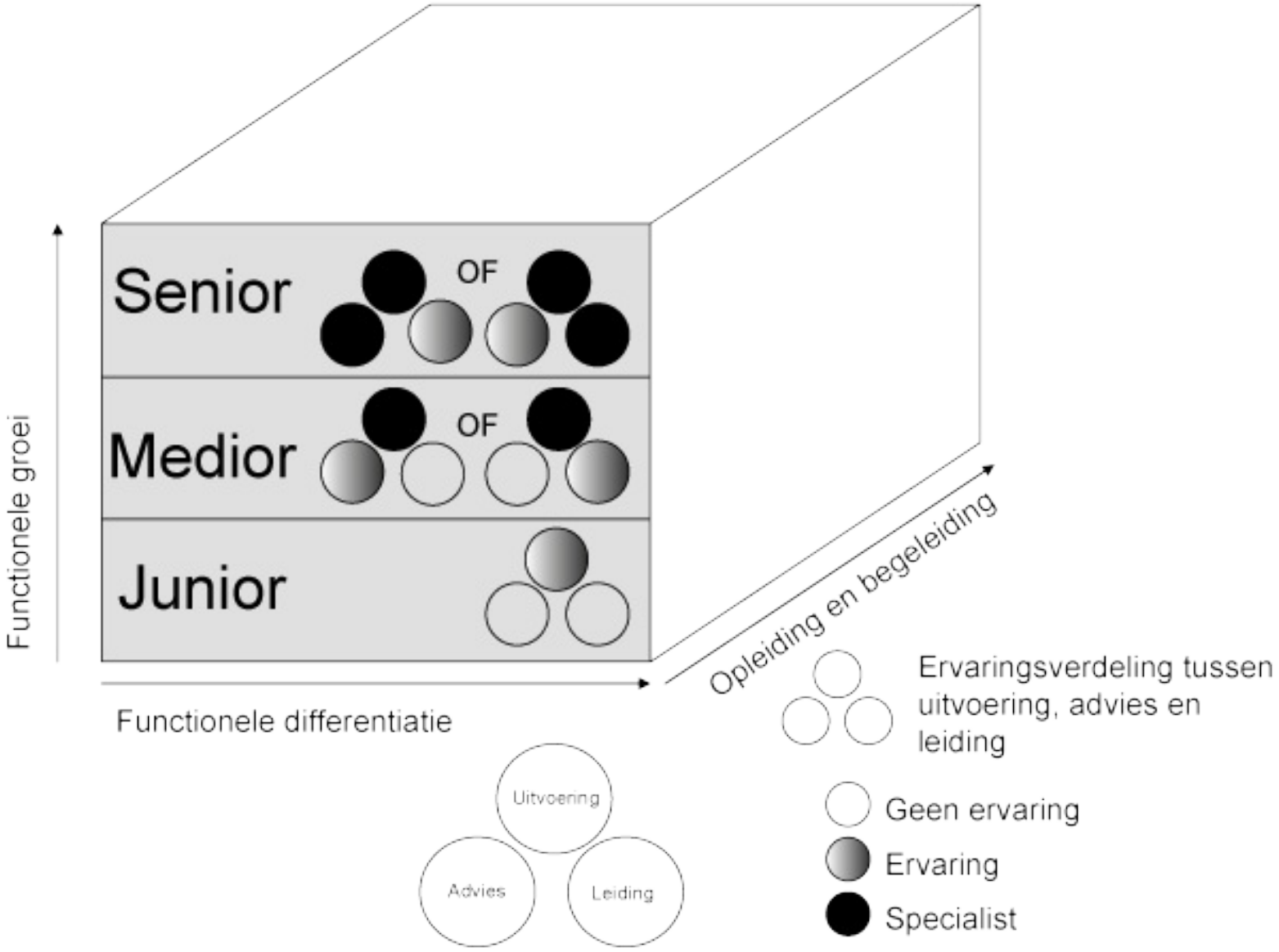
Onder functionele groei wordt het verticale functieverloop bedoeld dat een medewerker kan doorlopen. Globaal gezien bestaat dit uit drie hoofdgroepen van functies te weten junior, medior en senior. Het indelen naar deze drie groepen gebeurt op basis van het 3-rollenmodel. Hierin staat in welke 3 soorten rollen de testprofessional kan acteren binnen een testtraject en waar hij dus ervaring in heeft of misschien zelfs specialist is.

De drie mogelijke soorten rollen zijn:

- Uitvoerende rol;
- Adviserende rol;
- Leidinggevende rol.

Het onderscheid tussen junior, medior en senior wordt nu als volgt gemaakt (zie figuur 8.21 “De functionele groei in de carrièrekubus”):

- *Junior*: minimaal ervaring met de uitvoerende rol van testen;
- *Medior*: specialist op de uitvoerende rol van testen en minimaal ervaring met een adviserende rol of leidinggevende rol;
- *Senior*: specialist op de uitvoerende rol van testen en specialist op de adviserende rol of leidinggevende rol.



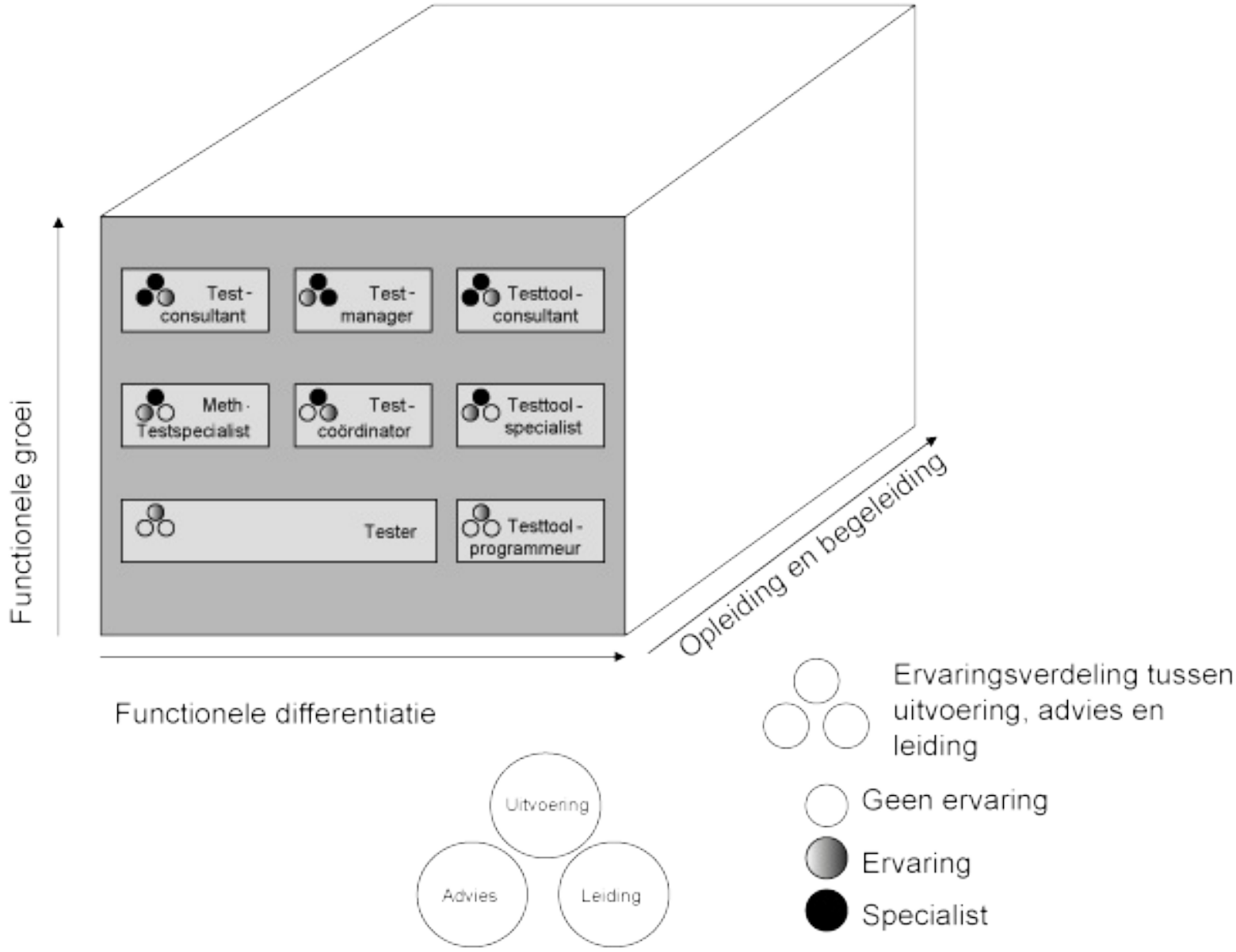
Figuur 8.21: De functionele groei in de carrièrekubus.

Tweede dimensie: functionele differentiatie

Vanuit TMap en de bijhorende carrièrepaden worden drie differentiaties in het vakgebied testen onderkend. De testprofessional kan zich specialiseren in een of meer van deze differentiaties. Deze drie differentiaties zijn:

- De uitvoering van het testproces;
- Het inzetten van tools voor het testproces;
- De aansturing van het testproces.

Deze drie differentiaties vormen samen met de 3 soorten rollen (uitvoerend, adviserend en leidinggevend) de verschillende functies die een testprofessional kan hebben. Gecombineerd met het 3-bollenmodel wat hiervoor is beschreven levert dat figuur 8.22 “De functionele differentiatie in de carrièrekubus” op.



Figuur 8.22: De functionele differentiatie in de carrièrekubus.

Derde dimensie: kennis en vaardigheden

Om de tester te ondersteunen in zijn functionele groei of differentiatie en om de aansluiting te houden bij de vraag vanuit de organisatie is het belangrijk om de juiste opleidingen en begeleiding aan te bieden. De opbouw van opleiding en begeleiding wordt gestalte gegeven vanuit de derde dimensie van de carrièrekubus.

De opleidingen dienen zich te richten op het vak, op gerelateerde vakgebieden (bv. kwaliteitszorg, requirements), op IT kennis (ontwikkelomgeving, talen), op vaardigheden (communicatie) en op materie kennis. De begeleiding richt zich op de houding en het toepassen van de kennis en ervaring. Juiste begeleiding is ook een voorwaarde voor een verantwoord carrièrepad. Elke medewerker krijgt op verschillende manieren ondersteuning. Dit kan zijn door een collega testprofessional en/of door andere ervaren medewerkers. Beginnende testers kunnen gecoacht worden door hun manager of een aangewezen coach. Naarmate het functieniveau toeneemt, neemt het belang van steeds weer nieuwe praktijkervaring en sociale vaardigheidstrainingen sterk toe ten opzichte van (test-) opleidingen, coaching en support. Vanuit de organisatie dienen de juiste zaken aangeboden te worden.

8.6.5 Functies

De mogelijke functies die een testprofessional kan hebben staan hieronder beschreven. De functies zijn gebaseerd op het model van de carrièrekubus zoals hiervoor is beschreven. De functies zijn als volgt verdeeld:
Junior functies:

- Tester

- Testtoolprogrammeur

Medior functies:

- Methodisch testspecialist
- Testcoördinator
- Testtoolspecialist

Senior functies:

- Testconsultant
- Testmanager
- Testtoolconsultant

Per functie wordt toegelicht wat de mogelijke taken zijn en welke kennis en vaardigheden zijn vereist.

Uitgediept

Functie van een testprofessional versus rol van een testprofessional

Het belangrijke verschil tussen een functie en een rol is dat de testprofessional een functie *heeft* en een rol *vervult*. Een functie van een testprofessional beschrijft formeel welke taken deze moet kunnen uitvoeren en welke kennis en vaardigheden deze moet hebben. De mogelijke functies die een testprofessional kan hebben binnen een organisatie liggen vast in de carrièrekubus zoals hiervoor beschreven. Een functie zegt iets over de volwassenheid van de testprofessional in het vakgebied testen. Een testprofessional wordt beoordeeld op basis van zijn functie.

Een rol is gericht op het vervullen van taken voor het testproject of de permanente testorganisatie in een specifieke situatie. Rollen worden vervuld door een testprofessional. De precieze invulling van de rol is situationeel afhankelijk en verschilt per project of organisatie. In sommige gevallen is de rol gelijk aan de functie (denk bijvoorbeeld aan de rol tester). Mogelijke rollen die een testprofessional kan hebben worden in hoofdstuk 16 “Testrollen” beschreven.

Voorbeeld 1

Een testproject wil de rol tester invullen. Daartoe wordt een persoon gezocht die de functie van tester heeft.

Voorbeeld 2

Een testproject zoekt invulling van de rol testteamleider met een persoon die ook moet gaan testen (meewerkend voorman). Daartoe wordt een iemand gezocht die de functie van tester heeft, beschikt over praktijkervaring als tester, beschikt over de capaciteit leiding geven en de wens heeft testteamleider te willen worden.

Tester

De Tester voert de primaire testtaken uit, ofwel de kern van het testproces. De testtaken zijn de meest creatieve, maar ook de meest arbeidsintensieve van de testactiviteiten. De functie Tester is in de TMap-fasering actief vanaf de intake van de testbasis in de fase Voorbereiding tot en met de conservering van de testware in de fase Afronding.

Mogelijke taken

- Intake van de testbasis (de specificaties);
- Specificatie van logische en fysieke testgevallen, uitgangsgegevens en testscripts;
- Vullen uitgangsbestanden;
- Uitvoeren van testgevallen (dynamisch testen);
- Beoordelen testresultaten;

- Uitvoeren van controles en onderzoeken (statisch testen);
- Registreren van bevindingen;
- Conserveren van testware.

Vereiste kennis en vaardigheden

Testspecifiek

- Globale kennis van de TMap-fasering;
- Vaardigheid in de intake- en testontwerptechnieken;
- Vaardigheid in toepassen van checklists in het kader van statische tests.

Automatisering

- Algemene automatiseringskennis en -ervaring;
- Globale kennis van systeemontwikkelmethoden;
- Vaardigheid in het interpreteren van de testbasis (requirements, functionele specificaties, enzovoort)

Algemeen

- Kennis van de materie en de lijnorganisatie;
- Het vermogen zich snel het gebruik van testtools eigen te maken;
- Kennis van projectorganisatie en een projectmatige werkaanpak;
- Vaardigheid in tekstverwerking en het werken met spreadsheets;
- Creativiteit en nauwkeurigheid.

Testtoolprogrammeur

De testtoolprogrammeur is verantwoordelijk voor het technische aspect van de geautomatiseerde testsuite. Vaak wordt deze gebouwd met tools voor geautomatiseerde testuitvoering (zie paragraaf 8.5.3 “Soorten testtools”). De testtoolprogrammeur vertaalt het ontwerp voor de testsuite naar te realiseren modules binnen de testtool. Hij realiseert binnen bestaande en nieuwe testtools de testsuite. Binnen de door de testtoolconsultant gegeven aanwijzingen, voert de testtoolprogrammeur het detailontwerp, de realisatie en implementatie van testsuites uit.

Mogelijke taken

- Realiseren, testen, beheren en onderhouden van een testsuite;
- Ondersteunen bij het ontwerp van een testsuite.

Vereiste kennis en vaardigheden

Testspecifiek

- Kennis van (de mogelijkheden van) tools voor geautomatiseerde testuitvoering;
- Kennis van automatisering van het testproces.

Automatisering

- Kennis van systeemontwikkelmethoden;
- Kennis van de architectuur en tools voor systeemontwikkeling;
- Kennis van hardware, software en datacommunicatiemiddelen;
- Ervaring in het gestructureerd programmeren (bij voorkeur in de programmeertaal van het tool).

Algemeen

- Goed analytisch denkvermogen;

- Creativiteit en nauwkeurigheid;
- Het vermogen zich snel het gebruik van testtools eigen te maken.

Methodisch testspecialist

De methodisch testspecialist is iemand die specialist is in het testen. Dit kan zijn in een speciale testvorm of -soort (bijvoorbeeld de acceptatietest of ketentest) of in een bepaalde omgeving. Hij is ervaren en vervult de combinatie van uitvoerende en adviserende rollen in projecten. Hij kan ook de rol meewerkend voorman van een team invullen.

Mogelijke taken

- De taken zoals die worden uitgevoerd door de tester;
- Ondersteuning bij het opstellen van (detail)testplannen;
- Ondersteuning bij de inrichting van de testorganisatie;
- Aannemen en beoordelen medewerkers vanuit het oogpunt van inhoudelijke testkennis;
- Uitvoeren werkverdeling en voortgangsbewaking;
- Ontwikkeling en onderhoud van testopleidingen;
- Doceren en coachen;
- Bemiddeling bij externe opleidingen;
- Adviseren en ondersteunen bij de toepassing van alle soorten testtechnieken;
- Adviseren van de testcoördinator;
- Verzamelen en evalueren van statistische gegevens over het testen.

Vereiste kennis en vaardigheden

Testspecifiek

- Uitgebreide, specialistische ervaring in het toepassen van de TMap-fasering en technieken;
- Algemene kennis over de inzet van testtools.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring;
- Specialistische kennis en ervaring in systeemontwerp en -realisatie;
- Kennis van architecturen en tools voor systeemontwikkeling;
- Algemene kennis van hardware, software en datacommunicatiemiddelen.

Algemeen

- Kennis van de materie en de lijnorganisatie;
- Kennis van projectorganisatie en een projectmatige werkaanpak;
- Creatief, nauwgezet en kan strikt methodisch te werk gaan;
- Goede contactuele eigenschappen en een motiverende uitstraling;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid.

Testtoolspecialist

Iemand die specialist is in het uitvoeren van de taken van “Testtoolprogrammeur”. Dit kan zijn voor speciale tools (bijvoorbeeld geautomatiseerde testuitvoering tool of performance-, load-, en stresstesttool) of in een bepaalde omgeving (interface testen, schermtesten). Hij is zeer ervaren en vervult de combinatie van uitvoerende en adviserende rollen in projecten. Hij kan ook de rol als meewerkend voorman van een team testtoolprogrammeurs invullen.

De testtoolspecialist helpt de testtoolconsultant bij de invoering van nieuwe tools in de organisatie.

Mogelijke taken

- De taken zoals die worden uitgevoerd door de testtoolprogrammeur;
- Ondersteunen bij het opstellen van de geautomatiseerde testsuite;
- Helpen bij uitvoeren quick scan bij invoeren testtools;
- Configureren van testtools;
- Aannemen en beoordelen medewerkers vanuit het oogpunt van inhoudelijke testtoolkennis;
- Uitvoeren werkverdeling en voortgangsbewaking;
- Ontwikkelen en onderhouden van toolopleidingen;
- Doceren en coachen;
- Bemiddelen bij externe opleidingen;
- Adviseren van de testcoördinator.

Vereiste kennis en vaardigheden

Testspecifiek

- Uitgebreide kennis van (de mogelijkheden van) verschillende tools voor uitvoering van de test;
- Uitgebreide kennis van automatisering van het testproces.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring;
- Kennis van de architectuur en tools voor systeemontwikkeling;
- Kennis van hardware, software en datacommunicatiemiddelen;
- Ruime ervaring in het gestructureerd programmeren (bij voorkeur in de programmeertaal van het tool).

Algemeen

- Goed analytisch denkvermogen;
- Creativiteit en nauwkeurigheid;
- Het vermogen zich snel het gebruik van testtools eigen te maken.

Testcoördinator

De testcoördinator geeft leiding aan een team testers, methodisch testspecialisten, testtoolprogrammeurs of testtoolspecialisten. De testcoördinator is verantwoordelijk voor de planning, de aansturing en de uitvoering van het testproces, binnen planning en budget met de juiste kwaliteit voor maximaal één testvorm of één testsoort. Hij rapporteert conform het testplan over de voortgang van het testproces en de kwaliteit van het testobject. De functie is actief gedurende de gehele TMap-fasering.

Mogelijke taken

- Opstellen, verkrijgen van goedkeuring en onderhouden van het testplan;
  - Risicoanalyse en strategiebepaling;
  - Opstellen planning en begroting;
  - Specificatie organisatie en infrastructuur;
  - Specificatie procedures, standaards en normen;
- Uitvoeren van het testplan binnen planning en budget;
  - Aannemen en beoordelen medewerkers;
  - Dagelijkse besturing van de testactiviteiten
  - Uitvoeren werkverdeling;
  - Vaststellen detailplannen;
  - Bewaking van de voortgang;
- Leiden intern overleg;
- Deelnemen aan projectoverleg;
- Opstellen vrijgave advies;
- Evaluatie testproces.

Vereiste kennis en vaardigheden

Testspecifiek

- Uitgebreide ervaring in de toepassing van de TMap-fasering en technieken;
- Ervaring als meewerkend voorman;
- Uitgebreide kennis over de inzet van testtools.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring;
- Goede kennis van systeemontwikkelmethoden;
- Kennis van architecturen en tools voor systeemontwikkeling;
- Algemene kennis van hardware, software en datacommunicatiemiddelen.

Algemeen

- Globale kennis van de materie en de lijnorganisatie;
- Uitstekende contactuele eigenschappen en een motiverende uitstraling;
- Uitstekende schriftelijke en mondelinge uitdrukkingsvaardigheid.

Testconsultant

De testconsultant is iemand die expert is in de uitvoering van de taken van methodisch testspecialist. Hij richt zich alleen nog maar op advieswerk rond de aspecten van testen. Hij ondersteunt het testproces op methodisch gebied in de meest ruime zin van het woord. De functie is ondersteunend voor alle testactiviteiten en voor alle leden van het testteam. Dus zowel voor het testmanagement of testcoördinatie bij bijvoorbeeld de strategiebepaling als voor de tester bij het specificeren van testgevallen. De functie kan actief zijn gedurende de gehele TMap-fasering.

Mogelijke taken

- Inrichten testtechnieken;
- Ontwikkelen van nieuwe testtechnieken;
- Opstellen testvoorschriftgeving;
- Ontwikkelen bijbehorende opleidingen;
- Doceren en coachen;
- Adviseren en ondersteunen bij de toepassing van alle soorten testtechnieken;
- Verbeteren van het testproces;
- Managementadvisering.

Vereiste kennis en vaardigheden

Testspecifiek

- Specialistische ervaring in het toepassen van de TMap-fasering en technieken;
- Uitgebreide kennis over de inzet van testtools;
- Kennis van het verbeteren van testprocessen.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring;
- Specialistische kennis en ervaring in systeemontwerp en -realisatie;
- Kennis van architecturen en tools voor systeemontwikkeling;
- Algemene kennis van hardware, software en datacommunicatie.

Algemeen

- Kennis van de materie en de lijnorganisatie;
- Het vermogen zich snel het gebruik van (test)tools eigen te maken;
- Kennis van projectorganisatie en een projectmatige werkaanpak;
- Creatief, nauwgezet en strikt methodisch te werk gaan;
- Goede contactuele eigenschappen en een motiverende uitstraling;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid.

Testtoolconsultant

De testtoolconsultant is iemand die expert is in het zijn van toolspecialist. Hij richt zich alleen nog maar richten op advieswerk rond de aspecten van testtools. Hij ondersteunt het proces rond de inzet van tools in de meest ruime zin van het woord. De functie is ondersteunend voor alle testtools en voor alle leden van het testteam.

Mogelijke taken

- Opstellen en inrichting testsuite;
- Ontwikkeling van nieuwe testsuites;
- Opstellen voorschriftgeving;
- Ontwikkeling bijbehorende opleidingen;
- Doceren en coachen;
- Adviseren en ondersteunen bij de toepassing van alle soorten tools;
- Opstellen toolbeleid;
- Uitvoeren van de quick scan bij het invoeren van tools;
- Managementadvisering.

Vereiste kennis en vaardigheden

Testspecifiek

- Specialistische kennis van (de mogelijkheden van) verschillende tools voor testuitvoering;
- Specialistische kennis van automatisering van het testproces.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring;
- Kennis van de architectuur en tools voor systeemontwikkeling;
- Kennis van hardware, software en datacommunicatiemiddelen;
- Ruime ervaring in het gestructureerd programmeren (bij voorkeur in de programmeertaal van het tool).

Algemeen

- Goed analytisch denkvermogen;
- Creativiteit en nauwkeurigheid;
- Het vermogen zich snel het gebruik van testtools eigen te maken;
- Goede contactuele eigenschappen en een motiverende uitstraling;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid.

Testmanager

De testmanager geeft leiding aan een team testcoördinatoren of testmanagers. De testmanager is verantwoordelijk voor de planning, de aansturing en de uitvoering van het testproces, binnen planning en budget met de juiste kwaliteit voor meerdere testvormen en testsoorten. Hij rapporteert conform het testplan over voortgang van het testproces en kwaliteit van het testobject. De functie is actief gedurende de gehele TMap-fasering.

Mogelijke taken

- De taken zoals die worden uitgevoerd door de testcoördinator;

- Opstellen, verkrijgen van goedkeuring en onderhouden van het mastertestplan;
  - Risicoanalyse en strategiebepaling;
  - Opstellen planning en begroting;
  - Specificatie organisatie en infrastructuur;
  - Specificatie procedures, standaards en normen.
- Uitvoeren van het mastertestplan binnen planning en budget;
  - Aannemen en beoordelen medewerkers;
  - Dagelijkse besturing van de testactiviteiten;
  - Inrichting organisatie;
  - De testfuncties en de procedures worden conform de in het mastertestplan gespecificeerde structuur geëffectueerd;
  - Uitvoeren werkverdeling;
  - Vaststellen mastertestplan;
  - Bewaking van de voortgang.
- Onderhouden externe contacten;
- Interne kwaliteitszorg;
- Onderkennen, anticiperen en rapporteren van projectrisico's;
- Definiëren van of adviseren over testbeleid;
- Uitvoeren optimalisatie testproces.

Vereiste kennis en vaardigheden

Testspecifiek

- Uitgebreide ervaring in de toepassing van de TMap-fasering en technieken;
- Ervaring als testcoördinator;
- Uitgebreide kennis over de inzet van testtools.

Automatisering

- Uitgebreide automatiseringskennis en -ervaring opgedaan;
- Goede kennis van systeemontwikkelmethoden;
- Kennis van architecturen en tools voor systeemontwikkeling;
- Algemene kennis van hardware, software en datacommunicatiemiddelen.

Algemeen

- Ruime ervaring in het leiding geven in projectverband;
- Globale kennis van de materie en de lijnorganisatie;
- Uitstekende contactuele eigenschappen en een motiverende uitstraling;
- Uitstekende schriftelijke en mondelinge uitdrukkingsvaardigheid.

8.6.6 Opleidingen

Zoals eerder beschreven, de eisen die aan testprofessionals worden gesteld zijn hoog. Van hen wordt verwacht zat ze kennis en ervaring hebben in:

- Het testvak;
- Gerelateerde vakgebieden;
- Algemene IT kennis;
- Sociale en communicatieve vaardigheden;
- De materie.

Het volgen van opleidingen is hier een vereiste voor. Een belangrijk instrument voor HRM is een opleidingsprogramma voor testmedewerkers. Dit kan gebruikt worden om het gat te bepalen tussen de aanwezige en gewenst kennis van de testprofessional. Het is raadzaam de testopleidingen modulair op te (laten) bouwen en beschikbaar te stellen. Afhankelijk van de functie en de bij de medewerker aanwezige expertise kan dan min of meer individueel per (deel)onderwerp worden opgeleid. Onderwerpen kunnen zijn:

Testvak:

- Testmethoden en technieken;
- Testomgeving en testtools;
- Testmanagement en beheer;
- Testprocesverbetering;
- Productrisicoanalyse en begroten;
- Testcertificering.

Uitgediept

Om grotere zekerheid te krijgen dat de testers voldoende testkennis hebben, kan specifiek gevraagd worden om gecertificeerde testers. Het EXIN organiseert een certificeringprogramma specifiek voor TMap. Daarnaast organiseert de ISTQB (International Software Testing Qualifications Board) een internationaal en algemeen certificeringprogramma voor testen.

Gerelateerde vakgebieden:

- Kwaliteitszorg;
- Requirements Lifecycle Management;
- Projectmanagement.

Algemene IT-kennis:

- Methoden en tools voor systeemontwikkeling;
- Plannings- en voortgangsbewakingstools;
- Systeembeheer en -exploitatie;
- Technische infrastructuur.

Sociale en communicatieve vaardigheden:

- Rapportage en documentatietechnieken;
- Testhouding;
- Sociale vaardigheden in woord, geschrift en persoonlijkheid;
- Effectief leiding geven.

Materie:

- Branchespecifieke materie en certificeringen;
- Algemene materie- en organisatiekennis.

Noten

1 Voor factoren die van buiten de organisatie een rol spelen wordt ook vaak de verzamelnaam “compliance” gebruikt.

2 Een ander veelgebruikte term is “dienstenniveau” of “prestatie-eis”

3 In de terminologie van beheer wordt vaak gewerkt volgens ITIL. Dit staat voor Information Technology Infrastructure Library en is ontwikkeld als een referentiekader voor het inrichten van een beheerorganisatie in de IT. Binnen ITIL wordt er gesproken over incidenten en niet over problemen. Voor meer informatie zie www.itil.com

4 Wordt vaak standaard meegeleverd met besturingssysteem.

5 Structured Query Language.

9 Productrisicoanalyse

9.1 Inleiding

Testen is een maatregel om inzicht te geven in de kwaliteit van opgeleverde producten en daaraan gerelateerde risico's bij het in productie nemen. Bij onvoldoende kwaliteit kunnen dan nog tijdig maatregelen genomen worden, zoals herstel door de ontwikkelaars. Er is echter nooit sprake van een onbeperkte hoeveelheid testmiddelen en -tijd. Daarom is het belangrijk om de testinspanning te relateren aan de verwachte productrisico's: grondig testen waar men grote risico's ziet, licht of niet testen waar de risico's klein zijn. Het motto hier is:

> "No risk, no test"

Hiervoor moeten onderbouwd keuzes gemaakt worden. Hulpmiddel bij het maken van deze keuzes is een productrisicoanalyse (PRA).

> **Definitie**
>
> De productrisicoanalyse is het analyseren van het te testen product met als doel dat testmanager en de verschillende andere belanghebbenden tot een gezamenlijk beeld komen van wat de meer of minder risicovolle kenmerken en delen van het te testen product zijn, zodat de grondigheid van testen hieraan gerelateerd kan worden.

De focus bij de PRA ligt op de productrisico's, ofwel wat is het risico voor de organisatie wanneer het product niet de verwachte kwaliteit heeft. Dit kan zijn omdat de software fouten bevat, maar ook omdat de AO-procedures niet aansluiten op het systeem of het systeem onvoldoende bruikbaar of te langzaam is voor de eindgebruikers.

> **Uitgediept**
>
> Daarnaast zijn er ook (test-)projectrisico's. Voorbeelden: het systeem móet op 1 januari in productie zijn, de oplevering van functionele specificaties is te laat, kundige testers komen niet beschikbaar of de testinfrastructuur is niet op tijd klaar. Deze risico's worden niet beschouwd bij het bepalen van de teststrategie, maar bij het opstellen van het testplan (zie paragraaf 5.2.11 en 6.2.13). Ook is op www.tmap.net een checklist met dergelijke risico's opgenomen, "Risico's testproces".

In de praktijk blijken mensen nogal breed uiteenlopende ideeën te hebben bij het begrip 'risico'. Bij een PRA is er behoefte aan een gemeenschappelijk referentiekader, met name wat onder een productrisico wordt verstaan.

> **Definitie**
>
> Een productrisico is de kans dat het product faalt in relatie tot de verwachte schade wanneer dit optreedt.
>
> Productrisico = Faalkans * Schade
>
> waarbij Faalkans = Foutkans * Frequentie van gebruik

De Faalkans is bepaald door de *foutkans* en de *frequentie van gebruik*. De foutkans is de kans dat een product(onderdeel) een fout bevat. Het aanwezig zijn van een fout in het product wil echter niet zeggen dat de fout zich ook manifesteert in productie. Wanneer de fout in een productonderdeel zit dat nooit gebruikt wordt, zal het product niet falen. Hoe vaker het productonderdeel met de fout gebruikt wordt, ofwel des te hoger de frequentie van gebruik, des te groter is de kans dat het product zal falen.

Om de analyse goed uit te kunnen voeren, worden daarom de afzonderlijke factoren van een risico beschouwd:

- Frequentie van gebruik
  Bij een systeem(deel) dat tientallen malen per dag aangeroepen wordt, is de kans dat een aanwezige fout zich manifesteert veel groter dan bij een systeem(deel) dat één keer per jaar wordt aangeroepen.
- Foutkans
  Om de foutkans in te schatten, kan het volgende overzicht van plaatsen waar de fouten zich vaak concentreren, behulpzaam zijn. Dit overzicht is mede gebaseerd op [Schaefer, 1996]:
  – Complexe functies (lees voor functies ook: functionaliteit, software);
  – Totaal nieuwe functies;
  – Veelvuldig aangepaste functies;
  – Functies waarbij bepaalde tools of technieken voor het eerst zijn ingezet;
  – Functies waarvan de ontwikkeling tussentijds aan een ander is overgedragen;
  – Functies die onder uitzonderlijk hoge tijdsdruk zijn gerealiseerd;
  – Functies waarbij meer dan gemiddeld geoptimaliseerd moet worden (bijvoorbeeld het heel snel of zuinig maken van een functie);
  – Functies waarin al eerder veel fouten zijn gevonden (bijvoorbeeld in een vorige release of tijdens eerdere toetsingen);
  – Functies met veel interfaces.
  De foutkans is ook groter bij:
  – Onervaren ontwikkelaars;
  – Onvoldoende betrokkenheid van gebruikers;
  – Onvoldoende kwaliteitszorg tijdens de ontwikkeling;
  – Onvoldoende kwaliteit van de ontwikkeltests;
  – Nieuwe ontwikkeltools en -omgeving;
  – Grote ontwikkelteams;
  – Ontwikkelteams met niet-optimale communicatie (door geografische of persoonlijke oorzaken).
  Belangrijk aandachtspunt ten aanzien van de foutkans (en daarmee de faalkans) is dat deze beter in te schatten is naarmate het ontwikkel- of onderhoudstraject vordert. Gedurende het testproces kan (en zal) de PRA daarom bijstelling behoeven.
- Schade
  Wanneer de fout zich manifesteert, wat is dan de schade die geleden wordt? Hierbij wordt een onderscheid gemaakt tussen directe en indirecte schade voor de organisatie. Directe schade is bijvoorbeeld omzetverlies, schade aan derden, economisch verlies, fysieke of milieuschade (speciaal bij embedded systemen: explosie, crash, verwonding), kosten voor correctie en herstel. Indirecte schade is bijvoorbeeld imagoschade, verlies van klantvertrouwen, schadeclaims van derden, overbelasting helpdesk of, bij overheidsinstanties, maatschappelijke schade zoals bij belasting (fouten bij belastingteruggave) en justitie (vormfouten waardoor zware criminelen niet veroordeeld worden). Bovendien kan sprake zijn van schade voor derden doordat het systeem niet werkt zoals bedoeld. De schade wordt meestal groter wanneer de fout doorwerkt in andere functies of systemen. Een organisatie kan op verschillende niveaus schade bepalen, zoals bijvoorbeeld op het niveau van bedrijfsprocessen, bedrijfsdoelstellingen, business requirements, maar ook op het niveau van systemen of deelsystemen.

9.2 Aanpak

Als regel vindt een PRA eerst plaats op mastertestplan-niveau, dus voorin het testtraject. Om de PRA zinvol te laten zijn, moeten het beschouwingsgebied van het testtraject en de business requirements al duidelijk zijn en moeten meer gedetailleerde requirements, ontwerpen of acceptatiecriteria al een eind op weg zijn.

Voor zover nodig worden aanvullingen en verdere detaillering gegeven op het niveau van de afzonderlijke testsoort. Is er geen mastertestplan, dan vindt een PRA op het niveau van de afzonderlijke testsoort plaats. Hierbij moet de testmanager er voor zorgen dat deze PRA enkel betrekking heeft op het beschouwingsgebied van de testsoort.

Het resultaat wordt opgenomen in het (master)testplan en vormt de basis voor de latere keuze in de strategie voor licht, zwaar of niet testen van een kenmerk (bijvoorbeeld een kwaliteitsattribuut) of deelobject (onderdeel) van het te testen product.

Men moet beseffen dat een PRA een momentopname is. Gedurende het verdere traject zal zeker blijken dat sommige risico’s overschat en andere onderschat of helemaal niet opgemerkt zijn. De testmanager moet dan de teststrategie en bijbehorende begroting en planning hierop bijsturen, zie ook paragraaf 5.3.4 en 6.3.4.

In deze paragraaf worden de stappen van een PRA beschreven en voorzien van aanwijzingen.

Stappen:

1. Bepalen deelnemers;
2. Bepalen van de PRA-aanpak;
3. Voorbereiden sessie/interviews;
4. Verzamelen en analyseren productrisico's;
5. Volledigheidscontrole.

Let op

Uit onderstaande beschrijving van activiteiten kan het lijken alsof het organiseren en (doen) uitvoeren van een PRA niet bijzonder moeilijk is. In de praktijk blijken hier echter veel testmanagers mee te worstelen. In dit hoofdstuk is geprobeerd het proces zo goed en helder mogelijk te beschrijven, maar benadrukt moet worden dat veel afhangt van de specifieke *testmanagementvaardigheden* van de testmanager. Dit betreft bijvoorbeeld communicatieve vaardigheden, diplomatie en diepgaande kennis van de mogelijke (te testen) risico's en kenmerken en het (ruw) kunnen inschatten van de kosten, tijd en praktische haalbaarheid om dit te testen.

Uitgediept

Iteraties

Wanneer gewerkt wordt met iteraties of incrementen, is de vraag wat de scope van de PRA is. Is dit het uiteindelijke te realiseren systeem of het tussenproduct na afloop van de iteratie of het increment? In zo'n geval gaat de voorkeur er naar uit om een eerste PRA uit te voeren met als scope het uiteindelijk te realiseren systeem, en daarna per iteratie of increment een kleinschalige, aanvullende PRA (mini-PRA) gericht op de iteratie/increment.

Voorbeeld: een systeem gaat bestaan uit 2 functionele deelsystemen. De PRA wijst uit dat naast functionaliteit ook performance en beveiliging moeten worden getest. In iteratie 1 wordt deelsysteem 1 gerealiseerd, in iteratie 2 komt daar deelsysteem 2 bij, vanaf iteratie 3 zijn performance en beveiliging testbaar. De mini-PRA en aansluitende teststrategie voor iteratie 1 bepalen dat deelsysteem 1 functioneel grondig wordt getest. De mini-PRA en teststrategie voor iteratie 2 bepalen dat deelsysteem 2 gemiddeld moet worden getest en deelsysteem 1 enkel licht op regressie. Voor iteratie 3 wordt besloten tot een functionele integratietest en regressietest plus tests van performance en beveiliging.

9.3 Bepalen deelnemers

Het inschatten van productrisico's is moeilijk. Er is kennis nodig van het systeem en de organisatie, van mogelijke schade en van de kans op fouten. De testmanager heeft deze kennis niet of slechts in beperkte mate. Bovendien is de kennis meestal over meerdere partijen of personen verspreid. De verantwoordelijkheid voor het goed inschatten van de productrisico's ligt daarom bij de organisatie en de opdrachtgever. In de praktijk is de testmanager meestal wel de facilitator en organisator van de PRA en benadert hiervoor diverse personen die kennis over de productrisico's kunnen inbrengen.

Als regel geldt dat de afnemende partijen het beste de schade en frequentie van gebruik kunnen inschatten, terwijl de leverende partijen het beste de foutkans kunnen inschatten. Onderstaand staan een aantal voorbeelden van respectievelijk afnemende en leverende partijen:

- Afnemers
  Opdrachtgever, (kern)gebruikers, lijnmanagers, stuurgroepleden, projectmanager en (functioneel en systeem-)beheerders kunnen gezien hun kennis van de omgeving waarin het product gebruikt gaat worden, inzicht geven in de schade bij falen en in de verwachte gebruiksfrequentie.
- Leveranciers
  Architecten, requirementmanagers, ontwikkelaars, ontwerpers, programmeurs, databaseadministrators (DBA'ers), kwaliteitszorgmedewerkers, testmanagers en projectmanager kunnen gezien hun kennis van het totstandkomingsproces en de technische werking een goede indicatie geven van de foutkans.

De testmanager maakt in deze stap de inschatting hoeveel en welke partijen en personen nodig zijn om een voldoende betrouwbare PRA te kunnen uitvoeren. Hierbij moet hij rekening houden met wat de testopdracht is, dus met het feit of het een PRA voor een mastertestplan betreft of een (aanvullende) PRA voor een systeem- of acceptatietestplan. Zo liggen voor de systeemtest de productrisico's meestal op het vlak van "doet het systeem wat er gespecificeerd is", terwijl voor een gebruikersacceptatietest de productrisico's meer spelen rond de vraag "ondersteunt het systeem de organisatie".

De te betrekken personen zijn niet noodzakelijkerwijs dezelfde personen die genoemd zijn bij de "Oriënteren opdracht"-activiteit (zie paragraaf 5.2.2 en 6.2.2). De PRA is bedoeld als een verdere uitwerking van de verkregen informatie uit de Oriëntatie-stap, maar is testspecifieker en meer gericht op volledigheid van de productrisico's rond het systeem.

Behalve dat de PRA de basis vormt voor de teststrategie, heeft het ook als voordeel dat de verschillende partijen zich beter bewust worden van zowel de risico's als de bijdrage die het testen kan leveren om deze risico's beter beheersbaar te maken. Er ontstaat een gezamenlijk en gedragen beeld van de belangrijkste risico's met bijbehorende classificatie. In het vervolg van het testtraject helpt dit om commitment te krijgen als er op dat terrein beslissingen genomen moeten worden.

Tips

- Door de complexiteit van de materie is het moeilijk om een PRA volledig objectief en gedetailleerd uit te voeren: het zijn globale inschattingen. De testmanager moet dit de deelnemers ook duidelijk maken.
- Bij een PRA vraagt de testmanager veel input van verschillende belanghebbenden.
  Door zoveel input te vragen kan bij sommige partijen het beeld ontstaan dat de testmanager niet in staat is zijn werk zelfstandig uit te voeren. De testmanager kan dit voorkomen door vooraf aan te geven in welke periode input wordt gevraagd en bovendien zoveel mogelijk voorstellen te doen die als uitgangspunt dienen.

9.4 Bepalen van de PRA-aanpak

De testmanager bepaalt op welke wijze de PRA gaat plaatsvinden. Deze stap gebeurt parallel aan of volgend op de vorige stap.

Bij het bepalen van de PRA-aanpak zijn eigenlijk twee belangrijke keuzes te maken:

- Op welke wijze wordt de PRA georganiseerd, in sessies of interviews?
- Welke risico-classificatie wordt gebruikt?

Onderstaand worden beide onderwerpen nader toegelicht.

9.4.1 Organisatie van de PRA

De PRA kan op de volgende manieren plaatsvinden:

1. De testmanager organiseert bij voorkeur één, maar zo nodig meerdere sessies met alle deelnemers;
2. De testmanager doet individuele interviews met de deelnemers in plaats van een sessie.
3. Er zijn diverse middenwegen tussen mogelijkheden 1 en 2. Zo kan de testmanager een sessie organiseren voor elke groep deelnemers (bijvoorbeeld gebruikers en IT'ers). Een andere variant is om interviews te doen met steeds 2 of 3 deelnemers in plaats van een sessie, waarbij de deelnemers complementair moeten zijn (bijvoorbeeld een gebruiker en een ontwikkelaar). Ook kan een sessie gehouden worden met de belangrijkste groep deelnemers en kunnen aanvullende interviews plaatsvinden met deelnemers met specifieke expertise.
   Omdat een PRA in complexere omgevingen veel mankracht kan vergen, is ook een tussenvorm om een eerste sessie te doen met een beperkt aantal deelnemers (liefst met goede kennis van het te bespreken product), waarna in een vervolgsessie een meer voltallig team de resultaten aanvult, corrigeert en goedkeurt.

Een sessie duurt maximaal een halve dag en een interview duurt maximaal 2 uur.

Uitgediept

Sessie of interview

In het algemeen heeft een sessie de voorkeur boven interviews, met name door het gezamenlijk commitment dat dit geeft. Toch zijn er ook goede redenen om interviews te houden. Onderstaand wordt een aantal aspecten gegeven dat van invloed is op de keuze:

- Behoefte aan gezamenlijk commitment
  Om de verschillende gezichtspunten van de deelnemers te laten zien en naar een gezamenlijke en gedragen visie toe te werken, heeft de sessie grote voorkeur boven interviews.
- (On)bekendheid met PRA en risico-gebaseerd testen
  Wanneer enkele deelnemers onbekend zijn met PRA en testen kan dit een sessie verstoren omdat telkens veel uitleg nodig is. Een voorafgaande kick-off sessie om de PRA toe te lichten is dan nodig. Wanneer dit niet mogelijk is, kunnen beter interviews gehouden worden;
- Aantal deelnemers
  Bij een grote groep deelnemers (> 8) neemt in een sessie de kans toe dat een verlegen iemand zijn zegje niet kan of wil doen. Splitsen over meerdere sessies of interviews is dan aan te bevelen;
- Politieke spanningen
  Wanneer dergelijke spanningen spelen, zijn er twee keuzes: of testmanager en opdrachtgever willen de spanningen zichtbaar en bespreekbaar maken of ze willen er omheen laveren. In het eerste geval is de sessie de beste keuze.
- Groepsdenken versus individueel denken
  Met sessies vindt kruisbestuiving van ideeën en gedachten plaats, waarbij het geheel meer is dan de som der delen. Aan de andere kant komen sommige individuen onvoldoende aan bod in sessies. Op basis van de te betrekken deelnemers moet de testmanager hier een keuze in maken.
- Weinig overhead versus gedegen aanpak
  Sessies geven naar de buitenwereld de indruk van een gedegen aanpak met (te) veel overhead, gezien het (grote) aantal betrokkenen dat een aantal uren bij elkaar zit. Wanneer de organisatie een grote voorkeur heeft voor “lean and mean” is dit een argument om interviews te houden;
- “Discussie-cultuur”
  Hoewel in een sessie altijd het risico bestaat dat dit verzandt in discussies over kleinigheden, is dit risico in sommige organisaties groter dan in andere. In die gevallen zijn interviews te prefereren.
- Persoonlijke voorkeur en vaardigheden
  Houdt de testmanager graag interviews of vindt hij het leuker een sessie te modereren? Een interview is makkelijker dan een sessie. Hoeveel ervaring heeft hij met dat laatste?

Kiest de testmanager voor een sessie, dan moet hij nog de keuze maken welke sessietechniek hij toepast. Technieken als Metaplan® of de gelijksoortige, maar wat minder formele brown-papersessies zijn hiervoor heel geschikt.

Uitgediept

Metaplan

Metaplantechnieken zijn hulpmiddelen om binnen korte tijd met een groep mensen ideeën te verzamelen. De methode is geïnitieerd door Eberhard Schnelle in Hamburg. Naast simpele visuele technieken, zoals het gebruik van borden met daarop kaartjes geplakt, bestaat de methode uit het gebruik van moderatoren, die de discussie faciliteren, en een gestructureerd proces van voorbereiding (waarbij een goede vraagstelling cruciaal is) tot en met afronding en evaluatie van de resultaten. Om een ervaren moderator te worden zijn vele jaren ervaring met metaplansessies nodig. Minder ervaren moderatoren (en testmanagers) passen de techniek in versimpelde vorm succesvol toe. Kenmerk van de methode is om ideeën vanuit verschillende invalshoeken te verzamelen, waarbij het categoriseren en prioriteren van de belangrijkste ideeën belangrijker wordt gevonden dan het nastreven van volledigheid. Bij een PRA kan een testmanager de metaplantechniek gebruiken om gezamenlijk de testdoelen te bepalen, de te testen kenmerk/deelobject-combinaties te inventariseren en de hierbij horende risico’s te classificeren.

Voor interviews is op www.tmap.net een checklist van een interview-agenda “PRA interview” opgenomen.

Voorafgaand aan en ook in de sessie of het interview kan inspiratie gehaald worden uit checklists, bijvoorbeeld “Checklist risicofactoren per kwaliteitsattribuut” (zie www.tmap.net) en ervaringen van vorige tests. Een aparte, formele aanpak is FMEA (Failure mode and effect analysis) waarbij de deelnemers voor alle functies van het product analyseren wat er fout zou kunnen gaan en wat dan de gevolgen zijn. Voor de belangrijke risico’s worden dan aanvullende

(test)maatregelen gedefinieerd. Voor meer informatie, zie www.fmeainfocentre.com.

Het inhoudelijke proces wordt verder besproken in paragraaf 9.6. Het resultaat hiervan wordt vastgelegd.

Uitgediept

Alternatief: PRA combineren met teststrategie

Wanneer het aantal deelnemers betrokken bij de PRA beperkt is en zij al eerdere ervaring met PRA hebben of de productrisico's vrij overzichtelijk zijn, kan het efficiënt zijn de PRA te combineren met de teststrategie (zie paragraaf 5.2.4 en 6.2.5) in één sessie.

Uitgediept

Kick-off sessie bij testen van onderhoud

Bij het inventariseren van de testbasis wil het nog wel eens voorkomen dat wijzigingen niet goed zijn gedocumenteerd en in het geval van ad-hoc onderhoud dat de concrete aanleiding/oorzaak niet benoemd is. Een beproefde manier om hier meer helderheid in te krijgen is het houden van een kick-off sessie met alle relevante partijen (functionele- en technische beheerders, ontwikkelaars, gebruikers, testers). In deze sessie wordt de productrisicoanalyse vooraf gegaan door een impactbepaling en gevolgd door het opstellen of aanpassen van de teststrategie met tevens aandacht voor niet-functionele kwaliteitsattributen. Houd bij het verzamelen van niet-functionele kwaliteitsattributen rekening met de bestaande situatie. Een aanscherping van bijvoorbeeld een performance-eis van drie naar één seconde is meestal lastig te realiseren door middel van onderhoud als daar bij het oorspronkelijke ontwerp van het systeem geen rekening mee is gehouden.

Specifiek in het geval van ad hoc onderhoud kan tijdens een kick-off sessie ter sprake komen hoe een fout in een testsituatie gereproduceerd kan worden.

De uitdaging bij het organiseren van een kick-off sessie bestaat uit het afstemmen van de agenda's van de gewenste deelnemers versus de beschikbare, meestal beperkte, testdoorlooptijd.

9.4.2 Bepalen classificatiewijze risico's

Om te bepalen wat de meer of minder risicovolle kenmerken en onderdelen van het te testen product zijn, is een manier van classificatie noodzakelijk waarmee aangegeven kan worden of iets een hoog of laag risico vormt. De verschillende risico's kunnen allemaal afzonderlijk worden gewaardeerd (absolute classificatie) of de verschillende risico's kunnen ten opzichte van elkaar worden gewaardeerd (relatieve classificatie). Deze twee manieren van risicoclassificatie worden hieronder nader toegelicht. In de meeste gevallen doet een organisatie er verstandig aan één risicoclassificatie-mechanisme te kiezen. Wanneer de risico's van de systemen zeer uiteenlopen, bijvoorbeeld bij administratieve en safety-critical systemen, is de classificatie echter beter toepassingsafhankelijk te maken.

Absolute classificatie

Bij deze manier van classificeren wordt voor elk risico afzonderlijk bepaald hoe groot de *schade bij falen* en hoe groot de *kans op falen* is. De grootte van de schade en de kans daarop worden tegen elkaar uitgezet. Hieruit blijkt de risicoklasse. Een voorbeeld hiervan is de onderstaande tabel [Broekman en Notenboom, 2003], waarbij A staat voor een hoog risico, B voor gemiddeld en C voor een laag risico.

		Kans op falen		
		Hoog	**Midden**	**Laag**
Schade bij falen	**Hoog**	A	B	B
	Midden	B	B	C
	Laag	C	C	C

Tabel 9.1: Absolute risicoclassificatie 3 × 3

De risicoklassen (A, B en C) in de bovenstaande tabel zijn niet symmetrisch verdeeld. Bij toepassing in de praktijk is gebleken dat veel organisaties het belangrijker vinden een risico te beheersen met een hoge schade en een lage kans op falen dan een risico met een lage schade en een hoge kans op falen. Het staat een organisatie natuurlijk vrij om de tabel naar eigen inzicht op te stellen.

Bij een klein testobject of bij een organisatie die nog weinig ervaring heeft met risicoanalyses is een systeem waarin alleen onderscheid wordt gemaakt in de categorieën Hoog en Laag veelal voldoende. De hierbij horende risicoklassen staan in de onderstaande tabel [Lyndsay, 2002]:

		Kans op falen	
		Hoog	**Laag**
Schade bij falen	**Hoog**	A	B
	Laag	C	C

Tabel 9.2: Absolute risicoclassificatie 2 × 2

Als mogelijke tussenstap om de schade bij falen en de kans op falen te bepalen, kunnen zogenaamde detailrisicofactoren worden gebruikt. Dit wil zeggen dat de schade bij falen en de kans op falen meer in detail wordt uitgewerkt. Hierbij is het van belang om vooraf vast te stellen hoe de score op de detailrisicofactoren uiteindelijk de risicoklasse bepaalt.

Enkele voorbeelden om dit te verduidelijken:

Voorbeeld

Voor systeem X geven de belanghebbenden aan dat het niet correct werken van functie Y een risico bij in-productie-name vormt.

Detailrisicofactoren ten aanzien van de schade bij falen voor functie Y zijn:

- De output is duidelijk zichtbaar voor de klant (schade H);
- De kans op verlies van imago is hoog (schade H);
- De doorwerking in andere functies is laag (schade L).

Detailrisicofactoren ten aanzien van de kans op falen voor functie Y zijn:

- De functie wordt heel vaak gebruikt (kans op falen H);
- De functie is eenvoudig (kans op falen L);
- De ontwikkelaars zijn erg ervaren (kans op falen L).

De belanghebbenden hebben de volgende afspraken gemaakt:

- Als er op 2 detailfactoren een H wordt gescoord is de waardering Hoog;
- Als er op 1 detailfactor een H wordt gescoord óf op twee detailfactoren een M dan is de waardering Midden;
- Anders is de waardering Laag.

Voor het bovenstaande voorbeeld betekent dit dat de Schade bij falen de waardering H (hoog) krijgt. De Kans op falen wordt in dit voorbeeld M (midden). Deze waarden resulteren in een risicoklasse B.

Voorbeeld

Bij een andere organisatie is de indeling in Schade specifiek gemaakt:

Hoog:	>250.000 euro
Midden:	tussen 50.000 en 250.000 euro
Laag:	< 50.000 euro

Relatieve classificatie

Bij deze manier van classificeren worden de verschillende productrisico’s ten opzichte van elkaar in een ‘volgorde van

belangrijkheid' geplaatst.

Met een aantal belanghebbenden worden de onderkende risico's geanalyseerd. Vervolgens wordt vastgesteld waar het risico ten opzichte van de overigen geplaatst moet worden. Zo ontstaat een opsomming van risico's, waarbij het belang van het risico de volgorde bepaalt.

Een voorbeeld om dit te verduidelijken:

Voorbeeld

Voor systeem X onderkennen de belanghebbenden de volgende risico's:

1. De performance van de nachtbatch is te laag waardoor de medewerkers 's ochtends geen toegang hebben tot het systeem.
2. De factuur die naar de klanten wordt gestuurd, bevat onjuiste bedragen.
3. De lijst met controletotalen ten behoeve van de interne accountantsdienst bevat onjuiste gegevens.

De belanghebbenden analyseren in een bijeenkomst de verschillende risico's en komen tot de volgorde 2, 1, 3. Zij zien risico 2 als het belangrijkste risico.

Deze manier van classificeren wordt in de praktijk veelal toegepast voor kleine testobjecten of in organisaties waar weinig ervaring is met het uitvoeren van risicoanalyses.

Tip

In een situatie waarin veel risico's worden onderkend (bijvoorbeeld 20), is het aan te bevelen geen volgorde te bepalen van 1 tot en met 20. Het gevaar is dat er teveel discussie ontstaat over de volgorde van de risico's. In zo'n geval kan beter gewerkt worden met een aantal groepen. Denk hierbij bijvoorbeeld aan drie groepen (hoog, gemiddeld en laag belang) waarbinnen de risico's worden geplaatst.

9.5 Voorbereiden sessie/interviews

Belangrijk is dat de testmanager de deelnemers goed informeert over wat de PRA inhoudt en wat er van hen verwacht wordt. Dit kan door elke deelnemer mondeling voor te lichten, door hen een toelichtende tekst toe te sturen of door eerst een gezamenlijke kick-off te organiseren. In zo'n sessie licht de testmanager kort de PRA, de teststrategie, de visie op testen en het komende testproces toe en kunnen de deelnemers vragen stellen. Hierbij moet de testmanager testjargon vermijden. Welk van deze manieren gekozen wordt, hangt met name af van de bekendheid van de deelnemers met PRA.

De testmanager vraagt ook aan de deelnemers om zich voor te bereiden op de sessie of interviews. Sommigen hebben geen voorbereiding nodig en kunnen de risico's snel benoemen, voor anderen is voorbereiding wel nuttig om hen alvast na te laten denken over de voor hen relevante risico's. De systeemdocumentatie en (bestaande en nieuwe) bedrijfs- en beheerprocessen zijn hierbij belangrijke input, zowel voor het verkrijgen van de testdoelen als voor het onderscheiden van de kenmerken en deelobjecten.

Vervolgens krijgen de deelnemers een uitnodiging toegestuurd voor de sessie of het interview. Vanzelfsprekend regelt (of laat regelen) de testmanager van te voren de hiervoor benodigde logistieke voorzieningen.

De testmanager bereidt zich voor door de gehanteerde begrippen in de organisatie, zowel business als IT, te kennen. Hij moet ervoor zorgen dat de kans op verwarring over terminologie en begrippen zo klein mogelijk is en dat de aanwezige deelnemers zoveel mogelijk hun eigen taal kunnen spreken. Veel testmanagers maken in de voorbereiding voor zichzelf al een eerste opzet van de PRA, om goed beslagen ten ijs te komen en om de echte PRA straks zo nodig op gang te krijgen of vlot te trekken. Toch is het aan te raden om deze eerste opzet zoveel mogelijk achter de hand te houden om de suggestie van voorkoken of manipuleren van het PRA-resultaat te vermijden.

Uitgediept

Indirecte PRA-input

Naast directe input voor de PRA zoals de testbasis, bedrijfs- en beheerprocessen, zijn er ook diverse informatiebronnen die indirecte input voor de PRA vormen:

- Opdrachtformulering
  De opdrachtformulering geeft het doel en de scope van het testen en geeft hiermee belangrijke aanwijzingen voor wat de opdrachtgever belangrijk vindt. Voor het MTP bevat het ook de voorziene testsoorten.
- Projectvoorstel en business case
  Dergelijke documenten geven vaak de redenen aan waarom het systeem of pakket ontwikkeld, geïmplementeerd of aangepast moet worden en geven soms op hoog niveau ook al risico-indicaties.
- Business requirements
  Requirements zijn onder te verdelen in business, user en system requirements. Requirements hebben een prioriteitsaanduiding. Zo'n aanduiding zegt echter niet zoveel over het ermee geassocieerde risico, hoogstens zegt het iets over de schade – en ook daar is discussie over mogelijk. Business requirements met bedrijfsdoelstellingen zoals X% sneller, goedkoper en/of meer omzet kunnen hoogstens als achtergrondinformatie worden meegenomen in de PRA of teststrategie. User requirements en system requirements zijn vormen van testbasis en zijn daardoor wel rechtstreekse input voor de PRA.
- Projectrisicoanalyse
  Vaak bevat een dergelijke analyse voornamelijk projectrisico's in plaats van productrisico's. Bovendien is er goede kans dat de risico's al met andere maatregelen afgedekt zijn wanneer de PRA start, waardoor de hoogte van het productrisico is afgenomen. Wanneer hier rekening mee wordt gehouden, is de projectrisicoanalyse een nuttige bron van informatie voor de PRA.
- Business-risico's
  Business-risico's zijn de risico's die spelen op het hoogste organisatieniveau, zoals het niet tijdig op de markt komen van een nieuw product, een succesvol product van een concurrent of tegenvallende verkoopcijfers. Deze risico's kunnen helpen bij de beeldvorming voor de PRA maar zijn geen rechtstreekse input. Testers zitten meestal niet op het juiste niveau in de organisatie om over business-risico's te praten. Ook zijn de risico's vaak niet gedocumenteerd, of in elk geval geen onderdeel van de testbasis. Toch is deze kennis heel waardevol om een goede risico-inschatting van het systeem te kunnen maken. Boodschap: zoek of vraag ernaar om beter begrip van het systeem te krijgen. Wanneer de business-risico's bekend zijn bij de PRA-deelnemers, wil dat niet zeggen dat de testmanager op dit niveau moet communiceren en rapporteren. Dat hangt helemaal af van het opdrachtgever-niveau en de testdoelen.

9.6 Verzamelen en analyseren productrisico's

De testmanager moet bij deze activiteit het uiteindelijke doel van de PRA niet uit het oog verliezen, namelijk dat bekend is:

- wat allemaal getest moet worden (deelobjecten);
- waar naar gekeken moet worden (kenmerken);
- hoe zwaar getest moet worden (risicoklassen).

Om dit resultaat te bereiken, is deze activiteit in een aantal deelactiviteiten onderverdeeld die onderstaand zijn beschreven:

1. Bepalen testdoelen
2. Vaststellen relevante kenmerken per testdoel
3. Onderscheiden deelobjecten per kenmerk
4. Per kenmerk invullen van de risicotabel voor testdoelen en deelobjecten, waarbij deze stap is onderverdeeld in de volgende substappen:
   a. Schade per testdoel
   b. Faalkans per deelobject
   c. Risicoklasse per testdoel/deelobject
   d. Bepalen risicoklasse per deelobject
5. Samenvoegen kenmerken/deelobjecten/risicoklassen in totaaloverzicht

Bovenstaande stappen kunnen de indruk geven van een enorme en complexe activiteit. Dit valt mee. De stappen zijn voor een beter begrip zo elementair mogelijk gehouden. Afhankelijk van de ervaring van de testmanager en van de deelnemers zullen bepaalde stappen zeer snel doorlopen en zelfs gecombineerd kunnen worden. Wel blijft overeind staan dat het

communiceren tussen de deelnemers de grootste uitdaging is.

1. Bepalen testdoelen

De testmanager wil toe naar een risico-inschatting voor deelobjecten en kenmerken. Hiervoor heeft hij input nodig van de verschillende deelnemers aan de PRA. In de praktijk blijkt echter het praten over deelobjecten en kenmerken in veel gevallen te technisch voor de deelnemers. De testmanager moet daarom de PRA starten vanuit de belevingswereld van de opdrachtgever en de andere acceptanten, ofwel vanuit datgene wat zij belangrijk vinden. Dit worden de testdoelen genoemd.

Definitie

Een testdoel is een voor de opdrachtgever relevant doel voor het testen, vaak geformuleerd in termen van door IT ondersteunde bedrijfsprocessen, gerealiseerde user requirements of use cases, kritische succesfactoren, wijzigingsvoorstellen of benoemde (af te dekken) risico's.

Als eerste worden daarom in een PRA de testdoelen van opdrachtgever en andere acceptanten verzameld en vastgesteld, voor zover dat nog niet is gebeurd in "Oriënteren opdracht", zie paragraaf 5.2.2 en 6.2.2. Onderstaande tabel geeft een aantal voorbeelden voor verschillende soorten testdoelen.

Soort testdoel	Voorbeelden
Bedrijfsprocessen	De processen facturering en orders blijven correct werken met de ondersteuning van het nieuwe/gewijzigde systeem.
Deelsystemen	Correct werkende deelsystemen Klantbeheer, Orders, maar soms ook op functie- of schermniveau: toevoegen klant, wijzigen klant.
Productrisico's	1. Klant krijgt incorrecte factuur, 2. Gebruikers kunnen niet met het pakket omgaan.
Acceptatiecriteria	Koppeling tussen nieuwe Klantsysteem en bestaande Ordersysteem werkt correct.
User requirements	Vaststellen kredietwaardigheid aanvrager gebeurt door BKR-toetsing.
Use cases	Opvoeren nieuwe klant, open rekening.
Kritische succesfactoren	Online hypotheekofferte moet binnen 1 minuut verschijnen op scherm van de tussenpersoon.
Kwaliteitsattributen	Functionaliteit, performance, gebruikersvriendelijkheid, inpasbaarheid.

Tabel 9.3: Voorbeelden testdoelen

Het is afhankelijk van de organisatie of project wat voor soort testdoelen men kiest. IT-gerichte organisaties of projecten kiezen vaak voor use cases, user requirements maar ook rechtstreeks voor deelsystemen of kwaliteitsattributen. Organisaties of projecten met veel gebruikersinbreng vinden het makkelijker te praten over bedrijfsprocessen, individuele productrisico's of over kritische succesfactoren. Bij onderhoud is bijvoorbeeld vaak een belangrijk testdoel dat de bestaande functionaliteit correct blijft werken, bij pakketimplementaties worden vaak processen als testdoelen gekozen. Het is zaak die testdoelen te kiezen die aanspreken bij de deelnemers.

Uitgediept

De testmanager moet vermijden dat er teveel testdoelen benoemd worden. Een globale vuistregel is tussen de 3 en 20. Wanneer meer testdoelen benoemd worden, wat met name kan gebeuren indien de deelnemers als testdoelen bepaalde af te dekken productrisico's kiezen, moet gekeken worden of deze afzonderlijke risico's in een beperkt aantal groepen ingedeeld kunnen worden

Het gebruik van testdoelen is een essentiële stap waarmee de testmanager de deelnemers als het ware aan de hand kan meenemen door de volgende stappen. Ook zal het later de brug naar de belanghebbenden (vanuit de business) kunnen slaan doordat het gerichte rapportage mogelijk maakt.

2. Vaststellen relevante kenmerken per testdoel

Vervolgens bepalen de deelnemers per testdoel wat de voor testen relevante kenmerken zijn. Met andere woorden, waar moet het testen allemaal naar kijken om later de testdoelen te kunnen halen.

Voor kenmerken worden meestal kwaliteitsattributen gebruikt. De TMap-set van kwaliteitsattributen is beschreven in

hoofdstuk 10. Wanneer de kwaliteitsattributen voor de deelnemers moeilijk te begrijpen zijn, kan de testmanager ook testvormen gebruiken, bijvoorbeeld installeerbaarheid, multi-user, regressie, enzovoort. Voor sommige deelnemers zegt een installatietest nu eenmaal meer dan het kwaliteitsattribuut continuïteit waartoe dit behoort. Op www.tmap.net is het overzicht, genaamd “Overzicht toegepaste testvormen”, te vinden waarin de diverse testvormen staan genoemd.

In de meeste gevallen is het kenmerk “functionaliteit” relevant. Het is aan de testmanager om in te brengen dat ook andere kenmerken als performance, gebruikersvriendelijkheid en beveiliging relevant voor testen kunnen zijn. Andersom zijn er ook kenmerken die juist *niet* getest hoeven te worden. Dit laatste kan zijn omdat ze geen risico vormen of omdat het kenmerk al door een andere test wordt afgedekt.

Soort testdoel	Voorbeelden	Voorbeelden van kenmerken
Bedrijfsprocessen	Facturering, orders	diverse: functionaliteit, gebruikersvriendelijkheid, performance, …
Deelsystemen	Deelsysteem Klantbeheer, Orders, maar soms ook op functieniveau: Toevoegen klant, wijzigen klant.	diverse: functionaliteit, gebruikersvriendelijkheid, performance, …
Productrisico’s	1. Klant krijgt incorrecte factuur 2. Gebruikers kunnen niet met het pakket omgaan	1. functionaliteit 2. gebruikersvriendelijkheid, inpasbaarheid
Acceptatiecriteria	Koppeling tussen nieuwe Klantsysteem en bestaande Ordersysteem werkt correct	functionaliteit, misschien performance en beveiliging
User requirements	Vaststellen kredietwaardigheid aanvrager gebeurt door BKRtoetsing	functionaliteit
Use cases	Opvoeren nieuwe klant, open rekening	diverse: functionaliteit, gebruikersvriendelijkheid, performance, …
Kritische succesfactoren	Online hypotheekofferte moet binnen 1 minuut verschijnen op scherm van de tussenpersoon	performance
Kwaliteitsattributen	Functionaliteit, performance, gebruikersvriendelijkheid, inpasbaarheid	1-op-1

Tabel 9.4: Voorbeelden kenmerken per testdoel

3. Onderscheiden deelobjecten per kenmerk

Na het kiezen van de voor testen relevante kenmerken per testdoel, worden de kenmerken verzameld. Voor elk kenmerk delen de deelnemers het testobject nu op in een aantal deelobjecten.

Definitie

Een deelobject is een, vanuit het te testen kenmerk bezien, logisch samenhangend deel van het testobject.

De opdeling in deelobjecten maakt het mogelijk later meer verfijning aan te brengen bij het kiezen van de testdekking. Belangrijk om op te merken is dat de opdeling in deelobjecten afhankelijk is van het kenmerk. Zo zal het kenmerk “functionaliteit” een andere opdeling in deelobjecten laten zien dan “performance” of “beveiliging”. In de praktijk is de opsplitsing in functionele delen (deelsystemen) het meest essentieel, aangezien hier straks de meeste testtijd in gaat zitten. Het is aan te bevelen de indeling overeen te laten komen met de ontwerp-indeling, tenzij er goede argumenten zijn om hier vanaf te wijken.

Voorbeeld

Kenmerk	Deelobjecten		
Functionaliteit	Deelsys1	Deelsys2	Totale systeem
Performance	Batch	Online	

Functioneel is het systeem op te delen in 2 grote deelsystemen, die afzonderlijk getest kunnen worden. Daarnaast is altijd nog een integrale test nodig op het totale systeem. Dit betekent een indeling in drie deelobjecten.

Bij het kenmerk Performance wordt onderscheid gemaakt in de online en batch performance, ofwel wat is de directe respons op een gebruikersactie en wat is de benodigde tijd voor het uitvoeren van batch-processen.

De opdeling in kenmerken/deelobjecten maakt het ook mogelijk om vroeg in het testproces sterk afwijkende risicodelen te kunnen onderkennen. Zo’n deel kan dan bijvoorbeeld als apart deelobject onderscheiden worden. Hiermee wordt een afwijkende benadering van testen voor verschillende onderdelen mogelijk.

Uitgediept

Bij een PRA als onderdeel van het mastertestplan is deze opdeling voor veel kenmerken soms nog niet te maken, omdat de benodigde informatie ontbreekt. In dat geval gaat de PRA tot het niveau van kenmerken en maakt het geen opdeling in deelobjecten. Dat komt dan later bij de testplannen van de afzonderlijke testsoorten.

Uitgediept

Er is geen richtlijn te geven in hoeveel deelobjecten een testobject opgedeeld moet worden. Een aantal tot 7 is goed beheersbaar, maar de praktijk laat ook situaties zien met tientallen deelobjecten, waarbij elke functie in het systeem als deelobject wordt behandeld. Hoewel veel werk, is het voordeel dat een zeer gedetailleerde risicoanalyse plaatsvindt en later heel snel uren en technieken per functie kunnen worden toegekend.

4. Per kenmerk invullen van de risicotabel voor testdoelen en deelobjecten

Per kenmerk worden de testdoelen uitgezet tegen de deelobjecten (functionele deelsystemen, performance online/batch, …) en bepalen de deelnemers de risicoclassificatie. Dit gebeurt in de hieronder beschreven substappen. In onderstaande voorbeelden wordt hiervoor absolute classificatie toegepast volgens tabel 9.1:

		Kans op falen		
		Hoog	**Midden**	**Laag**
Schade bij falen	**Hoog**	A	B	B
	Midden	B	B	C
	Laag	C	C	C

a. Schade per testdoel

Eerst schatten de deelnemers per testdoel in wat de schade is wanneer het kenmerk onvoldoende is. Hiervoor is met name gebruikersinbreng nodig. Onderstaand voorbeeld hanteert als testdoelen de te ondersteunen bedrijfsprocessen.

Kenmerk: Functionaliteit	
Testdoelen	**Schade**
Bedrijfsproces 1	H
Bedrijfsproces 2	H
Bedrijfsproces 3	L

b. Faalkans per deelobject

Vervolgens maken ze een inschatting van de faalkans voor elk deelobject. Deze inschatting vraagt meer inbreng vanuit de techniek en ontwikkelproces.

Functionaliteit	**Deelsys1**	**Deelsys2**	**Totale sys.**
Faalkans	H	M	L

c. Risicoklasse per testdoel/deelobject

Op de kruispunten van testdoelen en deelobjecten geven de deelnemers vervolgens met een risicoklasse aan of deze combinatie relevant is. De risicoklasse wordt afgeleid uit de schade per testdoel en de faalkans per deelobject. De deelnemers kunnen hiervan afwijken, bijvoorbeeld omdat zij het echte risico als veel groter of kleiner zien dan de risicoklasse die uit het product van faalkans x schade volgt. Een afwijking moet altijd toegelicht worden.

Kenmerk: Functionaliteit	**Deelobjecten:**	**Deelsys1**	**Deelsys2**	**Totale sys.**
	Faalkans	**H**	**M**	**L**
Testdoelen	**Schade**			
Bedrijfsproces 1	H	A	-*	-
Bedrijfsproces 2	H	A	A'**	C'
Bedrijfsproces 3	L	C	C	C

* Een streepje geeft aan dat de combinatie niet relevant is voor testen
** Met een notatie als ‘ (of hoofdletter of vet) kan worden aangegeven dat er iets speciaals over het kruispunt te melden is. Dit kan bijvoorbeeld zijn dat het een klein onderdeel van het deelsysteem betreft of dat er een afwijkend risico aan verbonden is dan op grond van faalkans/deelsysteem en schade/bedrijfsproces geconcludeerd kan worden.

d. Bepalen risicoklasse per deelobject

Het uiteindelijk benodigde resultaat dat in de risicotabel vastgelegd moet worden, is een risicoklasse per deelobject (van een kenmerk). Op basis hiervan kan in de teststrategie de keuze gemaakt worden de kenmerk/deelobjectcombinatie licht, gemiddeld of zwaar te testen. Deze risicoklasse is eigenlijk de optelsom van de risicoklassen op de kruispunten van het deelobject met de testdoelen in de voorgaande stap c. Hiervoor moeten afspraken gemaakt worden over hoe de risicoklassen van de afzonderlijke kruispunten gezamenlijk de risicoklasse van het deelobject bepalen.

De risicoklasse van het deelobject is in onderstaand voorbeeld bepaald door de volgende afspraak te hanteren: de risicoklasse van het deelobject als geheel is gelijk aan de risicoklasse van de kruispunten die het meeste voorkomt. Als risicoklasse A en C beide het meeste voorkomen, krijgt het deelobject risicoklasse B. Als risicoklasse A en B óf B en C het meeste voorkomen, dan wordt gekozen voor de hoogste risicoklasse.

Kenmerk: Functionaliteit	**Deelobjecten:**	**Deelsys1**	**Deelsys2**	**Totale sys.**
	Faalkans	**H**	**M**	**L**
Testdoelen	**Schade**			
Bedrijfsproces 1	H	A	-	-
Bedrijfsproces 2	H	A	A’	C’
Bedrijfsproces 3	L	C	C	C
Risicoklasse =>		A	B	C

Onderstaand een voorbeeld voor het kenmerk performance.

Kenmerk: Performance	**Deelobjecten:**	**Online**	**Batch**
	Faalkans	**M**	**L**
Testdoelen	**Schade**		
Bedrijfsproces 1	H	B	-
Bedrijfsproces 2	M*	B	C
Bedrijfsproces 3	M	-	C
Risicoklasse =>		B	C

* De schade wijkt af van de tabel erboven. Dit komt omdat de schade bij niet goed werkende functionaliteit groter wordt beoordeeld dan de schade bij onvoldoende performance.

Deze risicotabellen leggen de relatie tussen bijvoorbeeld testdoelen zoals bedrijfsprocessen, user requirements of afzonderlijke productrisico’s met de kenmerken en deelobjecten, zodat de testmanager in latere rapportages kan rapporteren op testdoel-niveau. Om die reden moeten de tabellen beheerd worden, ook na het opstellen van het testplan en dus gedurende het hele testproces.

Tips

Evenwichtige inschatting van risico’s

Aan een kenmerk/deelobject wordt een risicoklasse toegekend. Men doet dit op basis van het ingeschatte risico (schade x faalkans), aan de hand van testdoelen, kenmerken en deelobjecten. De testmanager moet echter zien te voorkomen dat elk kenmerk/deelobject de hoogste risicoklasse (A) krijgt toegewezen. De deelnemers/ belanghebbenden hebben een natuurlijke neiging de risico’s zwaar in te schatten en voor A te kiezen. Tips om dit te voorkomen zijn:

- uitgangspunt is dat alle schade- en faalkans-indicaties op “Midden” beginnen; de deelnemers moeten dan per indicatie “aantonen” waarom de schade of faalkans voor dit aspect hoger is dan gemiddeld.
- ga niet de kenmerken onderling vergelijken (“functionaliteit is risicovoller dan performance”), dan gaat een kenmerk als functionaliteit altijd A krijgen.
- bespreek de implicaties van een bepaalde keuze voor de testinspanning: wat zou een zware of juist lichte test op een bepaald kenmerk als allereerste indicatie bijvoorbeeld kosten in termen van tijd en geld.

Testprocesrisico's

Vaak worden in een PRA naast de productrisico's diverse andere soorten risico's genoemd, zoals proces- en (test-)projectrisico's. De testmanager moet erop toezien dat deze vastgelegd worden en doorgegeven aan de bevoegde partij of persoon, zoals de projectmanager. Typische testprocesrisico's worden in een aparte paragraaf in het (master)testplan opgenomen.

Communiceer in taal deelnemers

De testmanager moet erop toezien dat zoveel mogelijk wordt gecommuniceerd in de terminologie van de belanghebbenden. Heel simpel gesteld kunnen vaak de volgende vragen worden gesteld: "Wat als dit [testdoel/kenmerk/deelobject] niet werkt? En wat is dan het gevolg?". Een hierop aansluitende tip is dat de deelnemers moeten doorvragen tot het over tijd, geld, klanten of imago gaat. Dit betekent zo min mogelijk praten over producteigenschappen ("gevolg: factureringsprogramma heeft 4 uur nodig in plaats van 2 uur") maar over wat dat voor de organisatie betekent ("klant krijgt factuur pas na 2 dagen in plaats van de volgende dag").

Nieuwe requirements

De ervaring leert dat in een PRA nogal eens nieuwe requirements worden "ontdekt". De testmanager dient deze te verzamelen en terug te koppelen aan het project, de verantwoordelijke voor de requirements of de Change Control Board.

5. Samenvoegen kenmerken/deelobjecten/risicoklassen in totaaloverzicht

Van de afzonderlijke risicotabellen wordt vervolgens het PRA-totaaloverzicht opgesteld, zie hierna. Deze stap is een administratieve exercitie en vraagt geen inbreng van de deelnemers. Het voordeel van deze stap is dat de deelnemers, testmanager en opdrachtgever hiermee het totaaloverzicht krijgen. De risicotabellen worden optioneel als bijlage bij het (master)testplan opgenomen en in ieder geval beheerd om de latere rapportage over de testdoelen mogelijk te maken. De volgende tabel is een voorbeeld van een totaaloverzicht. In de argumentatie-kolom worden beknopt de, aan de testdoelen gerelateerde, redenen gegeven waarom het kenmerk of deelobject de betreffende risicoklasse heeft gekregen.

Kenmerk	Risicoklasse	Argumentatie
Functionaliteit		
- deelsys 1	A	hoge faalkans, wordt in cruciale processen 1 en 2 gebruikt
- deelsys 2	B	gemiddelde faalkans, slechts beperkt gebruikt in cruciale proces 2
- totaal	C	als deelsys 1 en 2 goed werken, is de kans op integratiefouten laag.
Gebr.vriendelijkheid	B	…
Performance		
- online	B	
- batch	C	
Beveiliging	A	
Inpasbaarheid	B	

Uitgediept

Bij onvoldoende detail

Indien ten tijde van het opstellen van het mastertestplan nog onvoldoende detail bekend is over bijvoorbeeld de deelobjecten, blijven in de eerste kolom alleen de kenmerken over, zie het voorbeeld hieronder.

Kenmerk	RK	Argumentatie
Functionaliteit	B	
Gebr.vriendelijkheid	B	
Performance	B	
Beveiliging	A	
Inpasbaarheid	B	
…		

De detaillering die nodig is voor het opstellen van een teststrategie moet dan later bij de PRA van de afzonderlijke testplannen verder uitgewerkt worden. Wanneer in bovenstaand voorbeeld functionaliteit en performance vanuit het mastertestplan worden toegewezen aan de systeemtest, zou de PRA hiervoor als volgt verder ingevuld kunnen

worden.

Voorbeeld systeemtest:

Kenmerk	RK MTP		Risicoklasse		Argumentatie
Functionaliteit					…
- Deelsys 1			A		
- Deelsys 2			B		
- Totaalsys			C		
Performance					
- online			B		
- batch			C		

RK MTP = Vanuit het mastertestplan toegekende risicoklasse aan het kenmerk
A = Hoge risicoklasse
B = Gemiddelde risicoklasse
C = Lage risicoklasse

Uitgediept

Productrisico's bij onderhoud

Aangezien er een (test)verschil is tussen nieuwbouw en onderhoud heeft dit gevolgen voor de uit te voeren stappen van de risicoanalyse.

Het voornaamste verschil tussen nieuwbouw en onderhoud voor de teststrategie is de *foutkans*. Dit betekent voor de PRA dat de risicoclassificatie van de kenmerken/ deelobjecten bij onderhoud anders komt te liggen dan bij een nieuwbouwsituatie.

Er wordt bij onderhoud een aantal aanpassingen, meestal naar aanleiding van bevindingen of wijzigingsvoorstellen, gedaan op een al bestaand systeem. Deze wijzigingen kunnen foutief geïmplementeerd zijn en dienen te worden getest. Bij het aanpassen is er ook een kleine kans dat er onbedoeld fouten in ongewijzigde delen van het systeem worden geïntroduceerd, waardoor het systeem in kwaliteit achteruit gaat. Dit verschijnsel van kwaliteitsachteruitgang heet regressie en is de reden dat ook de ongewijzigde delen van het systeem getest worden. Een kenmerk/deelobject dat bij de nieuwbouw een hoog risico had, blijft bij de onderhoudsrelease misschien dus wel ongewijzigd. Omdat de kans op regressie dan het enige risico is, hoeft er veel minder grondig getest te worden. Om die reden kunnen voor onderhoud de PRA en strategiebepaling worden aangepast door als deelobjecten de wijzigingen te onderscheiden. Een dergelijke vorm van PRA en strategiebepaling wordt vaak uitgevoerd met de in paragraaf 9.4.1 beschreven kick-off sessie. Per wijziging (een geaccepteerd wijzigingsvoorstel of een opgeloste bevinding) wordt geïnventariseerd welke kenmerken relevant zijn en welke delen van het systeem worden gewijzigd en welke delen van het systeem mogelijk worden beïnvloed door de wijziging.

Een tip hierbij is dat weliswaar de foutkans bij elke nieuwe release anders ligt, maar de schade en frequentie van gebruik veel minder snel wijzigen. Deze informatie is daarom goed te hergebruiken voor de PRA van elke volgende release. Wel dient de beheerorganisatie deze informatie dan actueel te houden.

9.7 Volledigheidscontrole

Het laatste onderdeel van de PRA is om te controleren of het resultaat van sessie(s) of interviews volledig genoeg is. Hiervoor kunnen testdoelen, kenmerken (denk aan de niet-functionele kwaliteitsattributen!) en/of deelobjecten gebruikt worden. Is een kenmerk nog niet geadresseerd, is dat dan omdat er nauwelijks risico aan verbonden is of omdat het vergeten is? Ook requirements kunnen hiervoor gebruikt worden, maar er moet op gelet worden dat dit er dan niet teveel worden. User requirements zijn goed beheersbaar. Business requirements zijn te abstract voor testen ("aanpassing in systeem moet 10% meer omzet genereren") en van system requirements zijn er waarschijnlijk teveel.

Bij het inventariseren van risico's per kwaliteitsattribuut kan gebruik worden gemaakt van het overzicht van kwaliteitsattributen in hoofdstuk 10 en de checklist 'Checklist risicofactoren per kwaliteitsattribuut'. Deze checklist staat op www.tmap.net.

De volledigheidscontrole is op twee manieren uit te voeren:

1. als onderdeel van de sessie;
2. wanneer hiervoor de tijd ontbreekt: in een aparte sessie met de 2 of 3 hoofd-belanghebbenden, met voorbereiding door de testmanager.

Het gehele PRA-resultaat wordt rondgestuurd naar de deelnemers en opdrachtgever ter goedkeuring en met de vraag of er nog aanvullende opmerkingen zijn. De testmanager neemt deze mee, tenzij hij verwacht dat dit veel discussie geeft. De opdrachtgever geeft de uiteindelijke goedkeuring en neemt bij eventuele discussiepunten een beslissing.

10 Kwaliteitsattributen en testvormen

10.1 Inleiding

In dit hoofdstuk worden eerst de binnen TMap gehanteerde kwaliteitsattributen benoemd en kort toegelicht. Vervolgens wordt een beperkt aantal specifieke testvormen besproken en wordt aangegeven wat hun relatie met deze kwaliteitsattributen is. Het betreft de testvormen voor het testen van:

- regressie
- usability
- performance
- portabiliteit
- informatiebeveiliging.

10.2 Kwaliteitsattributen

Voor het testen hanteert TMap de onderstaande set van kwaliteitsattributen. Een andere bekende set van kwaliteitsattributen is te vinden in de internationale standaard ISO9126. Het hanteren van een set kwaliteitsattributen, hetzij van TMap, hetzij ISO9126, is aan te raden als een soort volledigheidscheck. Zo kan gecontroleerd worden dat van alle te testen aspecten of kenmerken van een systeem of pakket een bewuste keuze is gemaakt deze wel of niet te testen. Het maakt hierbij niet veel uit welke set toegepast wordt. Vaak is hiervoor in de organisatie al een keuze gemaakt. Een afbeelding van de TMap kwaliteitsattributen op ISO9126 is te vinden op www.tmap.net.

Uitgediept

Er is een aantal redenen om de TMap set van kwaliteitsattributen te (blijven) hanteren en niet over te stappen op ISO9126:

- In veel organisaties is TMap de standaard voor testen, inclusief de TMap set aan kwaliteitsattributen. Deze organisaties hebben weinig behoefte om op een andere set kwaliteitsattributen over te stappen;
- Het testen van functionaliteit is één van de belangrijkste aandachtsgebieden voor testen en komt veel en vaak aan de orde in dit boek. ISO9126 ziet functionaliteit als een paraplubegrip waaronder ook bijvoorbeeld beveiliging en inpasbaarheid vallen. Binnen ISO valt onder het testen van functionaliteit dus ook het testen van beveiliging en inpasbaarheid. Dit is verwarrend in een testboek;
- ISO9126 is niet noodzakelijk beter of slechter dan de TMap set, het is alleen anders;
- Hoewel ISO9126 een internationale standaard is, blijkt in de praktijk dat veel organisaties weer hun eigen kleine variant hierop maken, wat afbreuk doet aan de kracht van ISO9126 als standaard. Ook hanteren diverse organisaties nog oude versies van ISO9126.

Onderstaande kwaliteitsattributen worden binnen TMap onderkend:

- beheerbaarheid
- beveiliging
- bruikbaarheid
- connectiviteit
- continuïteit
- controleerbaarheid
- flexibiliteit
- functionaliteit
- gebruikersvriendelijkheid
- herbruikbaarheid
- (geschiktheid) infrastructuur
- inpasbaarheid
- onderhoudbaarheid
- performance

- portabiliteit
- testbaarheid
- zuinigheid

Van elk kwaliteitsattribuut wordt hierna een beschrijving gegeven en wordt aangegeven op welke manieren het testen hiervan in de praktijk plaatsvindt, waarbij zo nodig wordt verwezen naar paragraaf 10.3 “Testvormen”. Ook is in paragraaf 6.2.8 “Toewijzen testeenheden en testtechnieken” een tabel opgenomen waarin voor een aantal kwaliteitsattributen bruikbare testontwerptechnieken worden genoemd.

Beheerbaarheid

Het gemak waarmee het informatiesysteem in operationele staat kan worden gebracht en gehouden.

Beheerbaarheid is primair gericht op het technisch systeembeheer. De installeerbaarheid van het informatiesysteem is een onderdeel van beheerbaarheid. Beheerbaarheid kan statisch worden getest door de aanwezigheid van maatregelen en hulpmiddelen die het beheer vereenvoudigen c.q. mogelijk maken, te beoordelen. Dynamisch testen van beheer gebeurt door bijvoorbeeld een installatietest uit te voeren en door de beheerprocedures (zoals backup en recovery) uit te voeren in de testomgeving.

Beveiliging

De zekerheid dat raadpleging of mutatie van de gegevens uitsluitend mogelijk is door die personen die daartoe bevoegd zijn.

In paragraaf 10.3 worden testvormen voor informatiebeveiliging besproken.

Bruikbaarheid

De mate waarin het informatiesysteem is toegesneden op de organisatie en het profiel van de eindgebruikers voor wie het bedoeld is, alsmede de mate waarin het informatiesysteem bijdraagt aan het bereiken van de bedrijfsdoelstellingen. Een bruikbaar informatiesysteem komt tot uiting in een verhoogde efficiëntie van de bedrijfsprocessen. Zal een nieuw systeem wel of niet functioneren in de praktijk? De enige die daar uiteindelijk antwoord op kan geven is de gebruikersorganisatie zelf!

Tijdens (gebruikers)acceptatietests wordt dit aspect normaal gesproken (impliciet) meegenomen. Indien het aspect bruikbaarheid expliciet wordt onderkend in de teststrategie, kan hiervoor een testvorm worden ingericht: de bedrijfssimulatie of proeftuin. Tijdens een bedrijfssimulatie test een aselecte groep potentiële gebruikers de bruikbaarheidsaspecten van het product in een omgeving die de “natuurlijke” omgeving waarin men het systeem gaat gebruiken, zoveel mogelijk benadert: de als-ware-het productie-omgeving. De test vindt plaats aan de hand van een aantal praktijkgerichte opdrachten c.q. testscripts. In de praktijk wordt het testen van bruikbaarheid vaak gecombineerd met het testen van gebruikersvriendelijkheid binnen de testvorm usability, zie paragraaf 10.3 voor meer informatie.

Connectiviteit

Het gemak waarmee een koppeling met een ander informatiesysteem of binnen het informatiesysteem tot stand kan worden gebracht en kan worden gewijzigd.

Connectiviteit wordt statisch getest door de betreffende maatregelen (zoals standaardisatie) met behulp van een checklist te beoordelen. Testen van connectiviteit gaat dus over het beoordelen van het gemak waarmee een (nieuwe) interface kan worden gelegd of gewijzigd en is dus niet het testen of een interface correct werkt. Dit laatste is normaal gesproken onderdeel van het testen van de functionaliteit.

Continuïteit

De zekerheid dat de gegevensverwerking ongestoord zal kunnen voortgaan, dat wil zeggen ook na ernstige storingen binnen redelijke termijn kan worden hervat. Het kwaliteitsattribuut continuïteit kan worden uitgesplitst in attributen die achtereenvolgens van toepassing zijn bij toenemende verstoring van het informatiesysteem:

- bedrijfszekerheid: de mate waarin de gegevensverwerking vrij blijft van storingen;
- robuustheid: de mate waarin de informatievoorziening gewoon door kan gaan nadat de storing is verholpen;
- herstelbaarheid: het gemak en de snelheid waarmee de informatievoorziening na een storing hersteld kan worden;
- degradatiemogelijkheid: het gemak waarmee de kern van de informatievoorziening kan worden voortgezet nadat een deel is uitgevallen;
- uitwijkmogelijkheid: het gemak waarmee (een deel van) de informatievoorziening op een andere locatie kan worden voortgezet.

Continuïteit kan statisch worden getest door de opzet en het bestaan van de maatregelen in het kader van continuïteit te beoordelen aan de hand van een checklist. Dynamisch impliciet testen is mogelijk door het opbouwen van statistieken tijdens het uitvoeren van andere tests. Het simuleren van langdurig systeemgebruik (bedrijfszekerheid) of het simuleren van uitval (robuust, herstelbaar, degradatie en uitwijk) zijn dynamische expliciete tests.

Controleerbaarheid

Het gemak waarmee de juistheid en volledigheid van de informatie (in de loop van de tijd) gecontroleerd kunnen worden. Bekende gebruikte middelen in dit verband zijn controletotalen, vierkants tellingen en audit-trail. Controleerbaarheid kan statisch worden getest gericht op de opzet van de betreffende maatregelen met behulp van een checklist en kan dynamisch expliciet worden getest gericht op de realisatie van de betreffende maatregelen in het systeem.

Flexibiliteit

De mate waarin de gebruiker zelf uitbreidingen of variaties op het informatiesysteem kan aanbrengen zonder dat de programmatuur wordt aan ge past. Oftewel: de mate waarin het systeem door de beheerorganisatie kan worden aangepast, zonder afhankelijk te zijn van de automatiseringsafdeling voor onderhoud.

Flexibiliteit wordt statisch getest door de betreffende maatregelen met behulp van een checklist te beoordelen. Dynamisch expliciet testen kan gebeuren door in de (gebruikers)acceptatietest de gebruiker bijvoorbeeld bij hypotheken een nieuwe hypotheekvariant te laten maken of bij creditcards de wijze van provisieberekening te laten aanpassen, in beide gevallen door parameters te wijzigen. Dit wordt vaak eerst op deze manier getest, voordat de wijziging daadwerkelijk in productie wordt doorgevoerd.

Functionaliteit

De mate van zekerheid dat de verwerking van de gegevens juist en volledig geschiedt.

Het kwaliteitsattribuut functionaliteit kan worden uitgesplitst in de attributen juistheid en volledigheid:

- juistheid: de mate waarin het systeem de aangeboden invoer en mutaties correct volgens de specificaties verwerkt tot consistente databases en uitvoer;
- volledigheid: de zekerheid dat àlle invoer en mutaties verwerkt worden door het systeem.

Bij het testen is het voldoen aan de gespecificeerde functionaliteit vaak het belangrijkste criterium om tot acceptatie van het informatiesysteem over te gaan. Met behulp van diverse technieken is de functionele werking dynamisch expliciet te testen.

Gebruikersvriendelijkheid

Het bedieningsgemak van het systeem voor de eindgebruikers. Vaak wordt deze algemene definitie uitgesplitst in: het gemak waarmee de eindgebruiker kan leren omgaan met het informatiesysteem en het gemak waarmee ingeleerde gebruikers met het informatiesysteem kunnen omgaan.

Voor gebruikersvriendelijkheid is een objectieve en werkbare meeteenheid moeilijk vast te stellen. Wel is het vaak mogelijk om in algemene bewoordingen een (subjectieve) mening te geven omtrent de gebruikersvriendelijkheid van het systeem.

Gebruikersvriendelijkheid wordt getest binnen de testvorm Usability, zie paragraaf 10.3 voor meer informatie.

Herbruikbaarheid

De mate waarin delen van het informatiesysteem, of van het ontwerp, opnieuw kunnen worden gebruikt voor de ontwikkeling van andere toepassingen.

Wanneer het systeem in hoge mate is gebaseerd op herbruikbare modules, komt dit ook de onderhoudbaarheid ten goede. Herbruikbaarheid wordt getest door het informatiesysteem en/of het ontwerp met behulp van een checklist te beoordelen.

(Geschiktheid) Infrastructuur

De geschiktheid van de apparatuur, het netwerk, de systeemsoftware, het DBMS en de (technische) architectuur in algemene zin voor de betreffende toepassing en de mate waarin deze infrastructuur-elementen op elkaar aansluiten.

Het testen van dit aspect kan op verschillende manieren. Van groot belang is hierbij de expertise van de tester op het gebied van de betreffende infrastructuur-elementen.

Inpasbaarheid

De mate waarin de handmatige procedures en het geautomatiseerde informatiesysteem op elkaar aansluiten en de werkbaarheid van deze handmatige procedures voor de organisatie.

Bij het testen van inpasbaarheid wordt vaak ook het aspect tijdigheid meegenomen. Tijdigheid wordt gedefinieerd als de mate waarin de informatie op tijd beschikbaar komt om de maatregelen te nemen waarvoor die informatie was bedoeld. Inpasbaarheid wordt dynamisch expliciet getest met behulp van de procescyclustest.

Onderhoudbaarheid

Het gemak waarmee het informatiesysteem kan worden aangepast aan nieuwe wensen van de gebruiker, aan de veranderende externe omgeving of om fouten te herstellen.

Inzicht in de onderhoudbaarheid wordt bijvoorbeeld verkregen door tijdens het testen de inspanning (in uren) te registreren die nodig is om een fout op te lossen of door te registreren hoe lang het gemiddeld duurt voor een bevinding is opgelost (Mean Time To Repair (MTTR)). Onderhoudbaarheid wordt tevens getest door de interne kwaliteit van het informatiesysteem (inclusief bijbehorende systeemdocumentatie) met behulp van een checklist te beoordelen. Inzicht in de gestructureerdheid van de programmatuur (een aspect van onderhoudbaarheid) wordt verkregen door het uitvoeren van statische tests, bij voorkeur ondersteund door codeanalysetools (zie ook paragraaf 7.2.8 “Testtools voor ontwikkeltests”).

Performance

De snelheid waarmee het informatiesysteem interactieve en batch-transacties afhandelt.

In paragraaf 10.3 worden testvormen voor performance besproken.

Portabiliteit

De diversiteit van het hardware- en softwareplatform waarin het informatiesysteem kan draaien, en het gemak waarmee het systeem kan worden overgebracht van de ene omgeving naar een andere omgeving.

In paragraaf 10.3 worden testvormen voor portabiliteit besproken.

Testbaarheid

Het gemak en de snelheid waarmee de functionaliteit en het prestatieniveau van het systeem (na iedere aanpassing) getest kunnen worden.

Testbaarheid betreft in dit geval het totale informatiesysteem. Van grote invloed op de testbaarheid van het systeem is de kwaliteit van de systeemdocumentatie. Dit wordt statisch gemeten met behulp van de checklist “Intake testbasis” tijdens de fase Voorbereiding. Voor het meten van de testbaarheid van het informatiesysteem kan eveneens een checklist gebruikt worden. Zaken die de testbaarheid (sterk) ten goede komen zijn:

- goede systeemdocumentatie;
- het hebben van een (geautomatiseerde) regressietest en andere testware;
- het gemak waarmee tussenresultaten van het systeem zichtbaar gemaakt, beoordeeld en zelfs gemanipuleerd kunnen worden;
- diverse testomgevingsaspecten, zoals representativiteit en een instelbare systeemdatum ten behoeve van tijdreizen.

Zuinigheid

De verhouding tussen het prestatieniveau van het systeem (uit te drukken in het transactievolume en de totale snelheid) en de hoeveelheid resources (CPU-cycles, I/O-tijd, geheugen- en netwerkbeslag, enz.) die daarvoor gebruikt worden.

Zuinigheid wordt dynamisch expliciet getest met behulp van tools die het te onderzoeken middelenbeslag meten en/of dynamisch impliciet door het opbouwen van statistieken (door diezelfde tools) tijdens het uitvoeren van tests gericht op functionaliteit. Met name bij embedded systemen speelt dit aspect vaak sterk.

10.3 Testvormen

In deze paragraaf wordt een aantal specifieke testvormen besproken. Behalve de regressietest betreffen dit testvormen voor andere kwaliteitsattributen dan functionaliteit. De reden hiervoor is dat deze testvormen in de praktijk steeds vaker voorkomen, maar dat voorbereiding, specificatie en uitvoering van deze tests andere soorten kennis vragen dan bij de functionele testvormen. Per testvorm wordt uitleg gegeven van het aspect waarop de testvorm zich richt, wat de relatie met de hiervoor beschreven kwaliteitsattributen is, wat het belang van de test is en wat voor testtechnieken mogelijk zijn.

Achtereenvolgens komen de volgende testvormen aan bod:

- regressie
- usability
- performance
- portabiliteit
- informatiebeveiliging.

10.3.1 Regressie

Wat is regressie?

Een systeem of pakket is vrijwel altijd aan veranderingen onderhevig. Wanneer het in productie is, wil de eigenaar ervan bepaalde wijzigingen of uitbreidingen doorvoeren. Maar zelfs al eerder, bij de nieuwbouw van een systeem of de implementatie van een pakket, worden aanpassingen gedaan. Dit betreffen dan meestal opgeloste bevindingen of geïmplementeerde wijzigingsvoorstellen. Bij iteratieve of agile ontwikkelmethoden is het telkens uitbrengen van nieuwe, uitgebreidere releases (ook wel increments genaamd) zelfs inherent aan de methode.

Bij het uitvoeren van de aanpassingen (of uitbreidingen) is het mogelijk dat onbedoeld fouten in ongewijzigde delen van het systeem (of pakket) worden geïntroduceerd, waardoor de kwaliteit van het systeem achteruit gaat. Dit verschijnsel van kwaliteitsachteruitgang heet regressie en is de reden dat ook de ongewijzigde delen van het systeem getest moeten worden. Hoewel regressie op alle kwaliteitsattributen betrekking kan hebben, is het testen ervan in de praktijk primair op functionaliteit gericht.

Definitie

Regressie is het verschijnsel dat de kwaliteit van een systeem als geheel achteruit gaat als gevolg van individuele aanpassingen.

Belang van regressietesten

De kans dat er fouten in een ongewijzigd onderdeel van het systeem zijn geslopen na een aanpassing, is kleiner dan wanneer het onderdeel nieuw gebouwd zou zijn. Vanuit de gedachte dat het risico bepaald wordt door de schade x faalkans, kan het testen van de ongewijzigde delen van het systeem daarom met relatief minder testinspanning plaatsvinden dan bij een nieuw of gewijzigd deel van het systeem. Dit betekent echter niet dat de regressietest weinig inspanning vraagt. Met name in onderhoudssituaties is de totale inspanning voor deze regressietest vaak veel groter dan de testinspanning die nodig is voor het gedetailleerd testen van de wijzigingen. De reden hiervoor is dat bij onderhoud meestal maar een zeer beperkt aantal functies wijzigt.

Definitie

Een regressietest is erop gericht om te controleren dat alle ongewijzigde onderdelen van een systeem nog correct functioneren na het doorvoeren van een wijziging.

Een goede regressietest is van onschatbare waarde. Zeker in de beheersituatie biedt de test het vertrouwen dat de nieuwe versie van het systeem of pakket als geheel nog goed werkt.

Testtechnieken

Voor regressietesten zijn geen vaste testontwerptechnieken voorgeschreven. Alle bestaande technieken kunnen worden gebruikt om de testgevallen in de test te specificeren. Wel richt een regressietest zich voornamelijk op de samenhang tussen de delen van het systeem, omdat hier de kans op regressie het grootst is. Dit betekent dat integratietestgevallen en ‘goed-pad’ testgevallen de voorkeur hebben boven testgevallen voor uitzonderlijke foutafhandelingssituaties. Vaak wordt de regressietest initieel gevuld met testgevallen uit de tests van nieuwe onderdelen of de oorspronkelijke nieuwbouwtests en later aangevuld met testgevallen om wijzigingen te testen. Geschikte testontwerptechnieken zijn bijvoorbeeld de datacombinatietest, de gegevenscyclustest en de procescyclustest.

Indien de productrisicoanalyse voor de nieuwbouw beschikbaar is, kunnen de hier toegekende schadefactoren aan kenmerken en deelobjecten een rol spelen bij de samenstelling van deze regressietest. Een regressietest kan beperkt of volledig worden uitgevoerd, afhankelijk van de risico’s én van de benodigde testinspanning. Voor een uitleg over de schaalbare regressietest wordt verwezen naar paragraaf 6.6 “Fase Specificatie”.

De regressietest wordt onderhouden door op basis van de wijzigingen in het systeem, zowel functionele aanpassingen als opgeloste fouten, de testset aan te passen of uit te breiden. Op deze manier blijft de regressietest voortdurend actueel.

Omdat de regressietest zich richt op het systeem als geheel, wordt de test heel vaak uitgevoerd (minimaal één keer voor elke release), terwijl de test inhoudelijk meestal maar weinig verandert. Dit in tegenstelling tot een test om een specifieke aanpassing te valideren: deze wordt meestal alleen voor de betreffende release uitgevoerd. De combinatie van een hoge gebruiksfrequentie en hoge stabiliteit betekent dat een goede herbruikbaarheid van de test erg belangrijk is. Het is dus belangrijk om een goed gestructureerde en gedocumenteerde testset op te stellen en te onderhouden.

Bij het uitvoeren van regressietests komt het gebruik van testtools voor geautomatiseerde testuitvoering goed tot z’n recht. Het grote voordeel van automatisering van de regressietest is dat tegen een geringe inspanning elke keer de volledige test kan worden uitgevoerd en geen keuze hoeft te worden gemaakt welk deel van de regressietest wel of niet uitgevoerd gaat worden.

10.3.2 Usability

Wat is usability?

Zoals voor de meeste IT-definities zijn er ook voor usability diverse definities te vinden. Zelfs de Internationale Standaards Organisatie hanteert twee definities:

Definitie

ISO 9241-11: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

ISO/IEC 9126: a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

Uit de verschillende definities komt een aantal aspecten naar voren die een rol spelen bij usability:

- Effectiviteit
  Kunnen gebruikers hun taak volbrengen en hun doel bereiken met het systeem?
- Efficiëntie
  Hoeveel moeite en tijd kost het de gebruikers om dit te doen?
- Tevredenheid
  Wat vinden de gebruikers van het bedieningsgemak van het systeem?
- Begrijpelijkheid
  Hoe makkelijk begrijpt de gebruiker wat het systeem aan invoer van hem verwacht en hoe begrijpelijk is de uitvoer voor deze?
- Leerbaarheid
  Hoe snel en makkelijk is de werking van het systeem te leren én te onthouden?
- Aantrekkelijkheid
  Hoe aantrekkelijk vindt de gebruiker het systeem? Denk bijvoorbeeld aan lay-out, kleurgebruik, plaatjes, filmpjes en interactie.
- Robuustheid
  Hoe makkelijk kunnen de gebruikers fouten met het systeem maken, hoe ernstig zijn deze en hoe makkelijk kunnen deze weer hersteld worden?

Wie de gebruiker in bovenstaande is, en welke taken deze wil uitvoeren, speelt een belangrijke rol. Gebruikers kunnen klanten van de organisatie zijn of gebruikers van het systeem binnen de organisatie, maar bijvoorbeeld ook systeembeheerders vallen hieronder. Ook moet men onderscheid maken in ongetrainde en onervaren gebruikers versus getrainde en ervaren gebruikers en in wat voor context het systeem gebruikt wordt. Een webapplicatie op een smart phone heeft andere normen voor usability dan een webapplicatie op de PC.

De TMap kwaliteitsattributen die het meest te maken hebben met usability zijn gebruikersvriendelijkheid en bruikbaarheid. Van dit laatste attribuut wordt bij usability testen gekeken naar de bruikbaarheid vanuit gebruikerstandpunt, niet naar de algehele bruikbaarheid voor de organisatie. Usability testen heeft ook enige overlap met attributen als performance (als het systeem niet snel genoeg is, verstoort dit de bruikbaarheid), functionaliteit (vaak is allerlei functionaliteit aan een systeem toegevoegd om het gebruikersvriendelijker te maken) en continuïteit (robuustheid tegen fouten).

Hoewel usability bij uitstek een subjectief begrip is, is er in de loop der tijd een groot aantal publicaties over het onderwerp verschenen. De bekendste persoon op dit gebied is ongetwijfeld Jakob Nielsen [Nielsen, 1999]. Daarnaast heeft het World Wide Web Consortium richtlijnen opgesteld voor toegankelijkheid van websites, zodat deze ook geschikt zijn voor visueel gehandicapte mensen [www.w3c.org].

Belang van usability testen

Ontegenzeggelijk is het belang van usability enorm toegenomen met de toenemende digitalisering en informatisering van de maatschappij. Organisaties hebben via internet nieuwe communicatiekanalen naar hun klanten en de markt gekregen, met nieuwe soorten diensten (online veilingen, instant prijsvergelijkingen). De website is daarmee de etalage en het visitekaartje van het bedrijf geworden. Usability wordt nog belangrijker wanneer de gebruiker voor dezelfde dienst of hetzelfde product tegen dezelfde prijs óf bij een concurrerende website kan afnemen, of bij een traditioneel communicatiekanaal zoals winkel of telefonische helpdesk.

Voorbeeld

Concurreren met traditionele communicatiekanalen

De overheid heeft een monopolie op het leveren van bepaalde diensten of informatie. Met webapplicaties kunnen forse kostenbesparingen gerealiseerd worden wanneer veel burgers hier gebruik van maken in plaats van de telefoon te pakken, formulieren in te sturen of naar het gemeentehuis te gaan. Als mensen echter niet met de website kunnen omgaan, zullen ze toch de traditionele kanalen blijven gebruiken.

Andere gevolgen van onvoldoende usability zijn dat de gebruikers:

- meer fouten maken, met allerlei herstelwerkzaamheden tot gevolg;
- door verwarring en meer knopbediening minder efficiënt werken;
- niet weten wat ze moeten doen en dus vaak de helpdesk bellen;
- lange inleertijd nodig hebben.

Voorbeeld

De usabillity van de PIN-automaat

De vroege pinautomaten in Nederland hadden de volgende volgorde in handelingen:

1. steek PIN-kaart in automaat
2. geef de PIN-code in
3. geef geldbedrag in
4. ontvang geld
5. haal PIN-kaart eruit

Het doel wat je hebt als gebruiker van een PIN-automaat, namelijk om geld te krijgen, is bij stap 4 gehaald. Regelmatig vergaten gebruikers dan ook hun kaart eruit te halen. Dit is inmiddels aangepast door de laatste twee stappen te verwisselen. Nu wordt eerst de kaart teruggeven en dan pas het geld. In andere landen, zoals de VS, is deze aanpassing nog niet geheel doorgevoerd, zoals één van de schrijvers tot zijn schade ondervond …

Hoewel de usability van websites de laatste jaren flink verbeterd is, blijft dit toch een risicofactor voor een succesvolle site. Ook laten de opkomst van de elektronische agenda's en smart phones usability problemen zien met websites voor mobiel gebruik. Overigens hebben usability problemen niet alleen betrekking op websites of maatwerkapplicaties, maar ook op bijvoorbeeld embedded software en standaard softwarepakketten. In dat laatste geval zijn de mogelijkheden om de usability nog te verbeteren echter vaak beperkt.

Testtechnieken

Voor het testen van usability staat een aantal technieken tot de beschikking. Een belangrijke constatering hierbij is dat in een laat stadium (zoals de acceptatietest) gevonden usability problemen vaak ingrijpend en moeilijk op te lossen zijn, bijvoorbeeld omdat de applicatienavigatie of alle schermbesturingen aangepast moeten worden. Usability én testen ervan moeten daarom vanaf het begin van het ontwerp aandacht krijgen, dan zijn nog relatief goedkoop aanpassingen te maken. Mogelijke testobjecten zijn bijvoorbeeld, naast het werkende systeem, prototypes en schermontwerpen. Op www.tmap.net is "Overzicht usability-technieken" opgenomen met daarin een groot aantal usability-technieken. Onderstaand staan enkele van de belangrijkste benoemd. Ruwweg hebben ze de volgende kenmerken:

- Moment van toepasbaarheid
  Kan de techniek al gebruikt worden voor schermontwerpen, is een werkend systeem nodig of is de techniek bedoeld voor een systeem dat al in productie is.
- Testers
  Wie beoordeelt de usability? Dit kunnen usability experts zijn en/of de feitelijke gebruikers.

Heuristic evaluation

Heuristic evaluation is één van de bekendste manieren om usability te testen.

Tijdens een heuristic evaluation wordt systematisch onderzoek gedaan naar de usability van het ontwerp van de gebruikersinterface. Het uiteindelijke doel van heuristic evaluation is om problemen in het design van de gebruikersinterface te ontdekken. Door dergelijke problemen al in het ontwerpstadium te ontdekken kunnen deze tijdig opgelost worden. Tijdens het proces van heuristic evaluation geeft een groep van 3-5 experts (evaluators) hun mening over de gebruikersinterface conform een aantal usability principes (ook wel de “heuristics” genoemd).

Uitgediept

Nielsen onderkent 10 heuristics, zie [Nielsen, 2006]:

- Zichtbaarheid van de systeemstatus
- Overeenkomst tussen systeem en de echte wereld
- Controle en vrijheid voor de gebruiker
- Consistentie en standaards
- Foutpreventie
- Herkenning in plaats van herinnering
- Flexibiliteit en efficiëntie van gebruik
- Esthetisch en minimalistisch ontwerp
- Hulp aan gebruiker om fouten te herkennen, oorzaak te achterhalen en te herstellen
- Help en documentatie

Usability test

De omgang van één of meer gebruikers met het systeem wordt in een gecontroleerde omgeving geobserveerd door een aantal waarnemers. Naast usability experts is ook aan te bevelen hier een aantal ontwerpers voor uit te nodigen. Er worden enkele door de gebruiker uit te voeren taken geselecteerd, die typisch zijn voor de toepassing.

Uitgediept

Een taakbeschrijving bestaat doorgaans uit:

1. Een schets van de uitgangssituatie, bestaande uit een beschrijving van de rol die de proefpersoon aanneemt en diens achtergrond, bijvoorbeeld een onervaren gebruiker of een ervaren beheerder.
2. Eén of meer opdrachten, bijvoorbeeld controleer de status van de laatste bestelling, vergelijk de prijzen tussen twee leveranciers en bestel een artikel bij de goedkoopste van de twee. De opdracht moet aangeven wát er moet gebeuren, maar niet hoe de gebruiker dit moet doen.

De proefpersonen dienen de beschrijving van de rol te lezen en zich voor te bereiden om vanuit die achtergrond de opdrachten uit te voeren.

Tijdens het uitvoeren van de opdrachten is het de bedoeling dat de proefpersoon voortdurend hardop zijn of haar gedachten uitspreekt en zegt wat hij of zij aan het doen is. Bijvoorbeeld een reactie kan zijn “Ik ga nu naar het menu en open de optie Informatie van bedrijf X, eens kijken of ik daar de routebeschrijving vind. Oh nee, hier staat het niet… (etc)”.

De waarnemers observeren het gedrag van de gebruiker en maken aantekeningen. In een zogenaamd usabilitylab bevinden de waarnemers zich achter spiegelglas en wordt alles opgenomen op video (zowel de beelden van de gebruiker als de beelden en handelingen op de computer). Ook een techniek als eye tracking (het registreren van de oogbewegingen op het scherm) en andere fysiologische metingen (hartslag, transpiratie) zijn hier mogelijk. Door de gebruikte infrastructuur en hulpmiddelen is een usabilitylab in de regel erg duur. Een goedkoper maar minder effectief alternatief voor een usabilitylab is om een waarnemer bij de gebruiker te laten zitten en bijvoorbeeld alleen een videocamera te gebruiken of de gebruikershandelingen op het systeem op te nemen.

Vervolgens beoordelen de waarnemer(s) de usability van het systeem aan de hand van bijvoorbeeld het aantal gemaakte fouten, de tijd om een taak te voltooien en het navigatiepad dat werd gevolgd. Ook gebruiken ze de opmerkingen van de deelnemers tijdens de test om de usability te beoordelen.

Vragenlijsten

Een andere manier van beoordelen is om met behulp van vragenlijsten de gebruikers te vragen om hun mening over het systeem te geven.

Hoewel ook toepasbaar op prototypes of zelfs schermontwerpen, worden vragenlijsten met name toegepast wanneer het systeem gereed, of zelfs al in productie is, Wanneer de deelnemers genoeg vragenlijsten hebben ingevuld, volgt een evaluatie van de resultaten. Hoewel het een relatief goedkope manier is om usability te testen, is het nadeel dat het resultaat geen bijzonder gedetailleerd beeld zal geven van wat er goed of fout is aan een systeem.

SUMI (Software Usability Measurement Inventory, http://sumi.ucc.ie) en WAMMI (Website Analysis and MeasureMent Inventory, www.wammi.com) zijn methoden die op het gebruik van vragenlijsten gebaseerd zijn.

Checklists, interviews

Een goedkope usability testtechniek is om tijdens het uitvoeren van andere (meestal functionele) tests tevens een usability checklist te hanteren. Nog een mogelijkheid is dat de testers en gebruikers na werken met het systeem geïnterviewd worden over hun beleving ervan.

Tools

Tenslotte zijn met name voor webapplicaties tools beschikbaar die allerlei controles kunnen uitvoeren. Voorbeelden van deze controles zijn:

- Hebben de plaatjes en animaties ook een voorziening die dezelfde informatie overbrengt (een tekstblok) voor het geval de plaatjes, animaties enzovoort niet werken? Dit laatste kan gebeuren als je een afwijkende browser gebruikt, geen videokaart hebt, of visueel gehandicapt bent.
- Is de omvang van plaatjes niet te groot waardoor de site te traag wordt?
- Bevat elke pagina een link om terug te keren naar de vorige pagina en/of een link om door te gaan naar de volgende pagina?
- Zijn de tekstblokken in een scrollveld niet te lang?
- Zijn alle links (nog) geldig?

10.3.3 Performance

Wat is performance?

Deze testvorm richt zich op de snelheid waarmee het testobject haar taken uitvoert voor diverse soorten verwerking zoals realtime, online en batch. (Sub-)vormen van performancetesten zijn load-, stress- of benchmarktesten. Deze testvormen hebben elk hun eigen definitie: loadtesten controleert of en hoe het testobject blijft presteren bij normaal en maximum verwachte bezetting, stresstesten controleert bij welke bezetting de performance van het testobject dramatisch gaat afnemen (het breekpunt), benchmarktesten voert dezelfde performancetest uit op verschillende configuraties van het te testen systeem en meet de resultaten. In de praktijk worden deze namen nogal eens door elkaar heen gebruikt, in dit boek wordt uitgegaan van de verzamelnaam performancetesten.

Om performance te kunnen testen moeten de eisen bekend zijn: wat is snel genoeg en onder welke omstandigheden is dat dan? In de praktijk blijkt het hier nogal aan te schorten en ontbreken harde eisen waar de performance aan moet voldoen en moet hier vanuit testen naar gevraagd worden. Daarom heeft de performancetester vaak een rol als katalysator bij het helder krijgen van deze eisen.

Voorbeelden van mogelijke eisen zijn:

- De 6-seconden regel (webpagina moet geladen zijn binnen 6 seconden);
- Binnen een uur moeten 200 hypotheken kunnen worden verwerkt;
- De webserver moet bij 1000 gebruikers pagina WX003 binnen 5 seconden hebben gegenereerd en verstuurd;
- Transactie A.01 moet binnen 20 seconden zijn verwerkt als er 100 gebruikers op de database actief zijn.

Deze eisen zijn geen van alle volledig. Altijd is extra informatie noodzakelijk over de situaties waaronder aan deze eis moet worden voldaan (database omvang / aantal gelijktijdige gebruikers / ingezette hardware). Geldt de eis bijvoorbeeld ook voor een gebruiker met een langzame internetverbinding, tijdens piekuren en wanneer andere applicaties actief zijn op dezelfde hardware?

Opgemerkt moet worden dat de totale performance vaak afhankelijk is van een groot aantal schakels, die zeker niet allemaal onder controle zijn te brengen: database omvang, netwerk, applicatie en bij internet de computer en verbinding van de gebruiker (waaronder de tussenliggende infrastructuur). Er zijn dan ook vele manieren om de performance te verbeteren. Performancetesten biedt informatie om te kiezen tussen het aanschaffen van extra hardware of softwarematige aanpassingen zoals het aanpassen van de software, het anders instellen van database- en netwerk-parameters en het optimaliseren van zware transacties.

Voorbeeld

Een website heeft een overzichtspagina met daarop alle te bestellen producten met korte toelichting en foto. Wanneer de gebruiker op een foto klikt, gaat hij door naar een detailpagina met daarop meer informatie over het product en een grote foto. De performance van de overzichtspagina was zeer slecht. Na onderzoek bleek waarom. De foto's op de overzichtspagina waren dezelfde als die op de detailpagina's. Dit waren omvangrijke foto's van hoge resolutie. Deze moesten voor de overzichtspagina allemaal ingeladen worden. De oplossing was om aparte, lage resolutie foto's te gebruiken voor de overzichtspagina.

Belang van performance, load en stress testen

De opkomst van browser-based systemen met grote aantallen gebruikers is een belangrijke reden voor organisaties om systemen op performance te testen. Ongeacht of de systemen via internet, intranet of extranet toegankelijk zijn, de communicatie tussen het systeem en haar gebruikers loopt over een groot aantal componenten, die elk bepaalde grenzen hebben wat betreft capaciteit en verwerkingssnelheid. Daarnaast zijn het onvoorspelbare aantal en gedrag van met name internetgebruikers van invloed op de performance. Vaak hebben organisaties tot hun spijt moeten constateren dat de toeloop op het systeem zo groot was, dat het systeem instortte. In diverse gevallen heeft dit ook het nieuws gehaald.

Wat hierbij gebeurt, is dat de responsetijden tot een bepaalde belasting slechts langzaam toenemen, maar na dit punt (het zogenaamde breekpunt) snel omhoog schieten. Zie ook de beschrijving van load profiles in paragraaf 14.3.8 "Statistisch gebruik: operational profiles en load profiles".

De vraag kan gesteld worden wat stresstesten nog toevoegt ten opzichte van een loadtest. Het antwoord hierop is dat de kennis over het breekpunt van het systeem of pakket de organisatie in staat stelt om een inschatting te maken of in de nabije toekomst en zo ja, wanneer dan, dit breekpunt bereikt gaat worden. De organisatie kan dan tijdig maatregelen nemen door bijvoorbeeld extra hardware aan te schaffen.

Behalve het internet is ook in allerlei andere gevallen performancetesten van belang. Dit kan zijn bij safety-critical embedded software die realtime, binnen een aantal milliseconden, moet reageren, tot zware batchprocessen die een maximale doorlooptijd van enkele uren mogen hebben.

Testtechnieken

Uitgangspunt voor de performancetest zijn de gestelde performance-eisen. Het gebrek hieraan kan resulteren in een ongestructureerde aanpak van testen en onduidelijkheid met betrekking tot de te behalen resultaten. Soms moet de tester zich dan beperken tot het meten van performance in plaats van het testen: "in die en die situatie is de responsetijd 20 seconden. We weten niet of dat acceptabel is of niet …".

Het testen van performance kan dynamisch impliciet door het opbouwen van statistieken tijdens het uitvoeren van andere tests. Statisch testen is mogelijk door de systeemcomponenten(configuraties) zoals DBMS en netwerk op performance door te laten rekenen door infrastructuurspecialisten. De grondigste vorm is het dynamisch expliciet testen, dat hierna verder wordt toegelicht.

Afhankelijk van de eisen kunnen de te testen situaties bijvoorbeeld als volgt opgebouwd worden:

- simulatie van één gebruiker/transactie;

- simulatie van het gemiddelde aantal gebruikers/transacties dat verwacht wordt;
- simulatie van het maximale aantal gebruikers/transacties dat het systeem aan moet kunnen;
- het systeem zwaarder belasten dan dit maximale gebruik, om te kijken bij welke belasting de responsetijden van het systeem sterk beginnen toe te nemen (breekpunt).

Een belangrijk onderdeel van performancetesten is dat er een evenwicht moet gevonden worden tussen representativiteit, risicovolle componenten en benodigde tijd en geld:

- Representativiteit
  Op welke manier kan het systeem zoveel mogelijk vergelijkbaar met de toekomstige productiesituatie belast worden;
- Risicovolle componenten
  Welke transacties of systeemcomponenten zijn als gevolg van requirements of hardwarelimieten het meest kwetsbaar bij toenemende load;
- Tijd en geld
  Binnen welk budget en doorlooptijd moet een oordeel gegeven worden over de performance.

Het technisch zo nauwkeurig mogelijk benaderen van de realiteit kost veel tijd en geld. Het focussen op de meest risicovolle componenten kan weer resulteren in een niet-representatieve situatie omdat bepaalde hoogvolume maar lage prioriteit transacties buiten de performance test worden gehouden.

De basis voor een gestructureerde performancetest wordt gevormd door het opstellen van respectievelijk een:

- Load-model;
- Iteratie-model;
- Meetplan.

Hierbij blijkt de inzet van testtools (load & stress, monitoring) nagenoeg onmisbaar. Het inzetten van een groot aantal “echte” gebruikers om een representatieve bezetting te verkrijgen lukt met veel inspanning nog wel één keer, maar meestal geen tweede keer. En de praktijk wijst uit dat deze tests juist periodiek herhaald worden, omdat bij onderhoud telkens weer aanpassingen in hard- en software plaatsvinden.

Uitgediept

Load-model

Het load-model beschrijft op logisch niveau hoe het testobject zal worden belast en hoe de performance zal worden gemeten. Het load-model is te vergelijken met het logisch testontwerp en bestaat uit de volgende onderdelen:

- Performance-eis(en)
  De eisen worden geïnventariseerd en zo nodig gedetailleerd. Een eis in termen van aantal gelijktijdige gebruikers is bijvoorbeeld onnauwkeurig als gevolg van een variabel werktempo en omdat niet bekend is hoe vaak een gebruiker een transactie uitvoert. Dus er is extra informatie nodig, bij voorkeur het aantal transacties/hits per tijdseenheid.
- Testobject
  Op basis van de eisen en de risico-inschatting besluit de tester wat het testobject is (het systeem als geheel of bijvoorbeeld alleen een database of applicatie server). Dit heeft een directe impact op de benodigde testomgeving (zie verderop).
- Te genereren belasting
  Het uiteindelijke doel van een load-model is om een representatieve belasting op het systeem te genereren (zie ook paragraaf 14.3.8 “Statistisch gebruik: operational profiles en load profiles”). Deze belasting bestaat uit een aantal gebruikers die elk bepaalde taken/transacties verrichten. De tester onderscheidt verschillende typen transacties (zoekopdracht, vullen bestelwagentje, vastleggen aankoop) en verschillende typen gebruikers (zoekers, kopers). Vervolgens bepaalt hij de frequentie waarmee de transacties uitgevoerd worden en het aantal benodigde gebruikers en relateert hij transacties en gebruikers aan elkaar tot een representatief beeld ontstaat.
  Voorbeeld
  Aantal transacties per uur:

transactie 1:	500 x startpagina
transactie 2:	50 x zoekopdracht / browse opdracht
transactie 3:	20 x vullen van bestelwagentje
transactie 4:	10 x vastleggen transactie

Voor het genereren van deze load kan gekozen worden voor 500 unieke gebrui kers (waarvan sommigen dus heel weinig handelingen zullen uitvoeren) maar het is ook mogelijk om slechts 50 actieve gebruikers te hebben (voor bijvoorbeeld zoekopdrachten en af en toe een keuze maken en bestellen) terwijl 50 andere gebruikers niets anders doen dan telkens de startpagina openen en weer sluiten, wat resulteert in het gewenste aantal transacties per uur.
- Testomgeving
  De te gebruiken testomgeving wordt bepaald op basis van het gedefinieerde testobject en de te genereren belasting.
- Indicatoren
  De te meten onderdelen zoals processorbeslag, geheugengebruik, diskbeslag en netwerkverkeer.
- Beschrijving vereiste uitgangssituatie
  De uitgangssituatie die nodig is om de test uit te kunnen voeren. Normaal gesproken moet deze wat betreft omvang representatief zijn voor de productiesituatie.
- Grafische weergave architectuur testobject
  Hoewel optioneel maakt dit de tests overzichtelijk en beter toegankelijk.

Met name bij het kiezen van het testobject, het bepalen welke transacties deel van de test gaan uitmaken en het definiëren van de testomgeving speelt de balans tussen enerzijds risico's en representativiteit en anderzijds tijd en geld een belangrijke rol. Hierbij moeten de kosten van tools die nodig zijn om grote aantallen gebruikers te simuleren en metingen te verrichten, niet onderschat worden.

Iteratiemodel

Het iteratiemodel beschrijft op technisch niveau hoe het testobject zal worden belast en hoe de performance zal worden gemeten. Een loadmodel beschrijft de gewenste belasting. In de praktijk moet die belasting worden bereikt via een spreiding van gebruikers die langzaam aan het gewenste verkeer gaan genereren. Indien alle gebruikers namelijk op hetzelfde moment starten, zullen heel lang de acties gelijk oplopen. Indien er bijvoorbeeld iets gedownload moet worden, gaan alle gebruikers dit op hetzelfde moment doen. Dit is geen realistisch gebruikersgedrag. Een iteratiemodel beschrijft de (technische) wijze waarop de belasting wordt 'uitgesmeerd', zodat snel een dynamische mix van acties wordt bereikt die vervolgens een bepaalde tijd kan worden gehandhaafd.

Een iteratiemodel is afhankelijk van het performancetesttool, aangezien het beschrijft hoe verschillende toolscripts en gebruikers ten opzichte van elkaar draaien.

Meetplan

Het meetplan geeft aan op welke onderdelen van de testomgeving gemeten wordt gedurende de testuitvoering. Hierbij wordt per onderdeel aangegeven welke metingen worden verricht, met welke intervallen en hoe de informatie na afloop geborgd en gepresenteerd wordt.

10.3.4 Portabiliteit

Wat is portabiliteit?

Van sommige systemen wordt geëist dat ze kunnen draaien op een aantal verschillende configuraties van hardware en (systeem)software. Het bekendste voorbeeld hiervan is dat een webapplicatie moet kunnen werken met verschillende browsers. Ondanks standaards waar webpagina's aan moeten voldoen, gaat elke browser hier anders mee om. Bovendien hebben de browsers een hoop instelmogelijkheden, die ook de werking van de webapplicatie kunnen verstoren.

Binnen TMap valt dit alles onder het kwaliteitsattribuut portabiliteit: de diversiteit van het hardware- en software-platform waarin het informatiesysteem kan draaien, en het gemak waarmee het systeem kan worden overgebracht van de ene omgeving naar een andere omgeving. Een andere term die in dit verband vaak gebruikt wordt is compatibiliteit. Compatibiliteit betreft het eerste deel van de definitie van portabiliteit: de mate waarin het informatiesysteem kan draaien in diverse omgevingen.

Belang van portabiliteitstesten

Wanneer een systeem portabiliteitsproblemen heeft op één of meer van de afgesproken configuraties is dit vaak te zien aan verminkte schermen of erger, aan het crashen van het systeem.

Wanneer we bij het voorbeeld van webapplicaties blijven, kan dit leiden tot ergernis voor de gebruiker, extra belasting van de helpdesk, verkeerde informatievoorziening en zelfs het mislopen van omzet omdat gebruikers geen bestelling kunnen doen.

Testtechnieken

Testen van portabiliteit gebeurt door een (beperkt) deel van de reeds uitgevoerde functionele tests te herhalen voor andere te ondersteunen configuraties. Welk deel van de tests herhaald moet worden, is een risicoafweging. Ook het gebruik van checklists is populair. Voor webapplicaties is een checklist beschikbaar op www.tmap.net, “Checklists internet testen”. Een aanwijzing is nog om bij het testen van verschillende browsers de standaard parameterinstellingen (kleuren, lettertypen, Java J/N) aan te passen. Verkeerde constructies zoals hard in de programmatuur gecodeerde instellingen worden dan sneller zichtbaar.

Het testen van portabiliteit als zodanig is daarmee eenvoudig maar arbeidsintensief. Wanneer een groot aantal configuraties moet worden ondersteund, kan dit het aantal uit te voeren tests doen exploderen. Wanneer bijvoorbeeld 6 browserversies en 4 besturingssystemen moeten worden ondersteund, levert dit al 24 mogelijke combinaties op. Om deze explosie te voorkomen, moeten alleen de belangrijkste combinaties getest worden. Dit kan worden gedaan door de te ondersteunen configuraties in een matrix uit te zetten. Verticaal komen dan de browserversies, horizontaal de besturingssystemen. Kruis dan de bij de gebruikers meest voorkomende combinaties aan, plus nog een aantal andere. De eis is dat elke browserversie en elk besturingssysteem minimaal één keer voorkomt, zonder dat elke combinatie browser/OS getest hoeft te worden. Laat bij de keuze ook meespelen of problemen verwacht worden. In onderstaande tabel is een voorbeeld van een dergelijke matrix gegeven. De ‘X’-en stellen de te testen configuraties voor.

	Win 2000	WinXP	MacOS9	MacOS X
Internet Explorer 6	X	X	X	
Internet Explorer 7		X	X	X
Opera 8.5		X		
Firefox 1.5		X	X	X

Tip

Voor dit soort situaties zijn de dekkingsvormen/basistechnieken “equivalentieklassen” en “pairwise testing” bij uitstek geschikt, zie ook paragraaf 14.3.4 “Equivalentieklassen” en paragraaf 14.3.5 “Orthogonale arrays en Pairwise testing”.

Ook kan het lastig zijn om de juiste configuraties te verkrijgen. Zo is het een hele speurtocht om nog een PC met Windows2000 en een Netscape browser te regelen en daarna nog lastiger om deze aangesloten te krijgen op het bedrijfsnetwerk. Hoewel vaak op één machine meerdere configuraties geïnstalleerd kunnen worden (dual-boot, meerdere browsers), kan het ondersteunen van een groot aantal configuraties een grote investering in infrastructuur betekenen. Om dit alles te regelen, moet vroeg begonnen worden met de voorbereidingen.

Tip

Bij onderzoeksbureaus zijn lijsten op te vragen met de meest gebruikte software/hardware combinaties. Deze kunnen dan dienen als input voor de keuze in configuraties.

10.3.5 Informatiebeveiliging

Wat is informatiebeveiliging?

Informatiebeveiliging richt zich op het garanderen van de beschikbaarheid, exclusiviteit en integriteit van alle vormen van informatie met als doel de continuïteit van de organisatie te waarborgen en de eventuele gevolgen van

beveiligingsincidenten tot een acceptabel niveau te beperken. Naast het TMapkwaliteitsattribuut beveiliging houdt informatiebeveiliging zich ook expliciet bezig met aspecten als beschikbaarheid (continuïteit), integriteit (juistheid, compleetheid) en vertrouwelijkheid (exclusiviteit).

Het inrichten van informatiebeveiliging bestaat uit het nemen van afdoende maatregelen (gericht op bovengenoemde criteria) en het instellen van een kwaliteitsevaluatiemechanisme (Information Security Management System).

Het is een complex vakgebied dat niet onderschat mag worden. Zo zijn er op netwerk-niveau 8 dimensies van beveiliging te onderscheiden (ISO/IEC 18028-2) waarvoor maatregelen genomen kunnen worden tegen bedreigingen:

- Autorisatie
  regelt de toegang tot functies, applicaties, netwerk, enzovoort;
- Authenticatie
  bevestigt de identiteit van de communicerende partij;
- Onweerlegbaarheid
  legt een audit trail aan zodat de oorsprong van gegevens of de oorzaak van een gebeurtenis of actie later niet ontkend kan worden;
- Data confidentialiteit
  beschermt gegevens tegen ongeautoriseerde inzage
- Communicatie beveiliging
  regelt dat informatie alleen tussen geautoriseerde punten beweegt zonder omgeleid of onderschept te worden;
- Data integriteit
  bewaakt de correctheid van de gegevens en beschermt tegen ongeautoriseerde creatie, aanpassing, verwijdering of replicatie;
- Beschikbaarheid
  verzekert dat geautoriseerde toegang tot netwerkonderdelen, opgeslagen informatie, informatiestromen, services en applicaties niet geweigerd wordt;
- Privacy
  beschermt informatie die uit het observeren van netwerkactiviteiten gehaald kan worden.

Op technisch niveau zijn dan nog de volgende lagen van beveiliging te onderscheiden:

- Netwerk & infrastructuur
  Deze laag bestaat uit elementen zoals servers, firewalls, gateways, proxy servers en routers, zowel hardware als software. Een goede (beveiligings)arch itectuur is nodig om deze elementen zo met elkaar in verband te brengen dat een goede balans ontstaat tussen een gelaagde beveiliging en de wens om de applicaties toegankelijk te maken voor diverse partijen.
- Systeemsoftware
  Hieronder vallen zaken als het operating systeem en de DBMS. Toegangscodes, wachtwoorden en autorisaties kunnen hier worden beheerd. De onderdelen in deze laag hebben vaak allerlei instelmogelijkheden om de beveiliging te verhogen. Helaas blijken die in de praktijk vaak onvoldoende gebruikt te worden.
- Applicaties
  Dit zijn de applicaties waar de gebruikers toegang toe hebben, al of niet over het internet. Beveiligingsaspecten zijn bijvoorbeeld internet-cookies (kleine bestandjes met informatie die bij de gebruiker neergezet worden) met onversleutelde informatie, ongespecificeerde en onbeschermde navigatiemogelijkheden waardoor een gebruiker bij onderdelen van de applicatie kan komen waar deze niet geautoriseerd voor is, (on)versleutelde gegevens, onveilige scripts, niet valide invoer aan het systeem waardoor de applicatie crasht of in een instabiele, onvoorziene toestand terechtkomt.

Het mag duidelijk zijn dat voor informatiebeveiliging een groot scala aan maatregelen nodig is, zowel preventief als detectief. Ook kent een goede informatiebeveiliging meerdere lagen, zodat doorbreken van de eerste laag niet gelijk leidt tot het volkomen openstaan van alle gegevens. Bij soorten maatregelen moet men denken aan speciale tools (firewalls, virusdetectieprogramma’s, encryptieprogramma’s), instellingen van hard- en softwarecomponenten, normen en standaards voor systeembeheer, maar ook voor systeemontwikkeling, procedurele maatregelen en fysieke beveiligingsmaatregelen. De ontwikkelingen rondom informatiebeveiliging gaan snel. Bovendien moet in de mate van beveiliging altijd een balans worden gevonden met andere aspecten als toegankelijkheid, gebruikersvriendelijkheid en snelheid. Een organisatie doet er daarom goed aan periodiek het beveiligingsbeleid tegen het licht te houden: is de actuele mate van beveiliging nog wel in overeenstemming met de gewenste mate van beveiliging.

Belang van testen informatiebeveiliging

Informatiebeveiliging heeft sinds enkele jaren een enorme vlucht doorgemaakt. Dit wordt veroorzaakt door:

- media-aandacht voor en schade door hackpogingen;
- schandalen rondom de boekhouding van grote organisaties;
- eisen (veiligheidsgaranties) gesteld door leveranciers en afnemers;
- de beschikbaarheid van relevante normen (ISO2700x, NEN7510);
- diverse wetgeving voor organisaties rondom interne controles, financiële rapportering, verspreiding van informatie en bescherming van persoonsgegevens.

Daarmee wordt ook het testen van informatiebeveiliging(smaatregelen) een steeds belangrijker vakgebied. Het actueel houden van de kennis op dit gebied vraagt een bijna continue investering. Voortdurend worden nieuwe beveiligingslekken gevonden die door kwaadwillende hackers kunnen worden geëxploiteerd. www.tmap.net bevat een aantal links naar sites die de laatste informatie op dit gebied geven.

Wat en wanneer testen?

Er zijn diverse redenen om het niveau van informatiebeveiliging te testen (zie opsomming in de vorige paragraaf), waarvan de belangrijkste reden het verkrijgen van inzicht in het correct functioneren van de reeds ingevoerde maatregelen is. Dit is immers essentieel om er zeker van te zijn dat de genomen maatregelen voldoen aan de eisen die de organisatie stelt. Ook certificerende instanties geven hierover een oordeel, waardoor organisaties een normaccreditatie kunnen verkrijgen ten aanzien van bepaalde compliance- of wetgevingsissues.

Het moment waarop men tot testen overgaat, is divers. Dit wordt veroorzaakt door de status waarin een organisatie zich bevindt op het moment dat men de aandacht op informatiebeveiliging richt. Geen organisatie is hetzelfde en informatiebeveiliging is altijd op een bepaald niveau aanwezig (netwerk- en applicatie-login, firewalls, badges enzovoort) Er is ook een onderscheid te maken tussen deze beveiliging van de organisatie en het beveiligen van de individuele applicaties die de organisatie beheert, verkoopt of ter beschikking stelt aan leveranciers en afnemers al dan niet via een webtoepassing.

Bij de eerstgenoemde beveiligingsomgeving concentreren de maatregelen zich vooral op de beschikbaarheid en vertrouwelijkheid van informatie. Bij de applicaties zijn deze uiteraard ook van belang, maar is juist de integriteit van de informatie ook een issue. De testaanpak voor die verschillende kwaliteitsaspecten is ook anders.

Testen van beschikbaarheid

Bij het aspect beschikbaarheid wordt met name getest of hardware en software (waarop de informatie aanwezig is) operationeel is op de door de organisatie gewenste momenten. Hierbij moet onder andere gedacht worden aan de bescherming tegen Denial of Service-aanvallen. Hierbij proberen hackers een site onbereikbaar te maken door deze te overladen met valse informatieaanvragen.

Deze tests zijn veelal het aandachtsgebied van IT-afdelingen, aangevuld met specialisten op het gebied van (het verderop uitgelegde) penetratietesten en infrastructuren. Netwerk- en infrastructuur testen wordt steeds vaker gezien als een aandachtsgebied dat ook geschikt is om via (aanpassingen op) standaard testmethoden als TMap aan te pakken.

Testen van vertrouwelijkheid en integriteit

Testen van vertrouwelijkheid en integriteit van informatie gaat dan om vragen als: “wie mag toegang tot” en “hoe staat het met de integriteit van de ingevoerde gegevens”. Dit wordt aangevuld met “is reeds eerder ingevoerde informatie (alsnog) manipuleerbaar”.

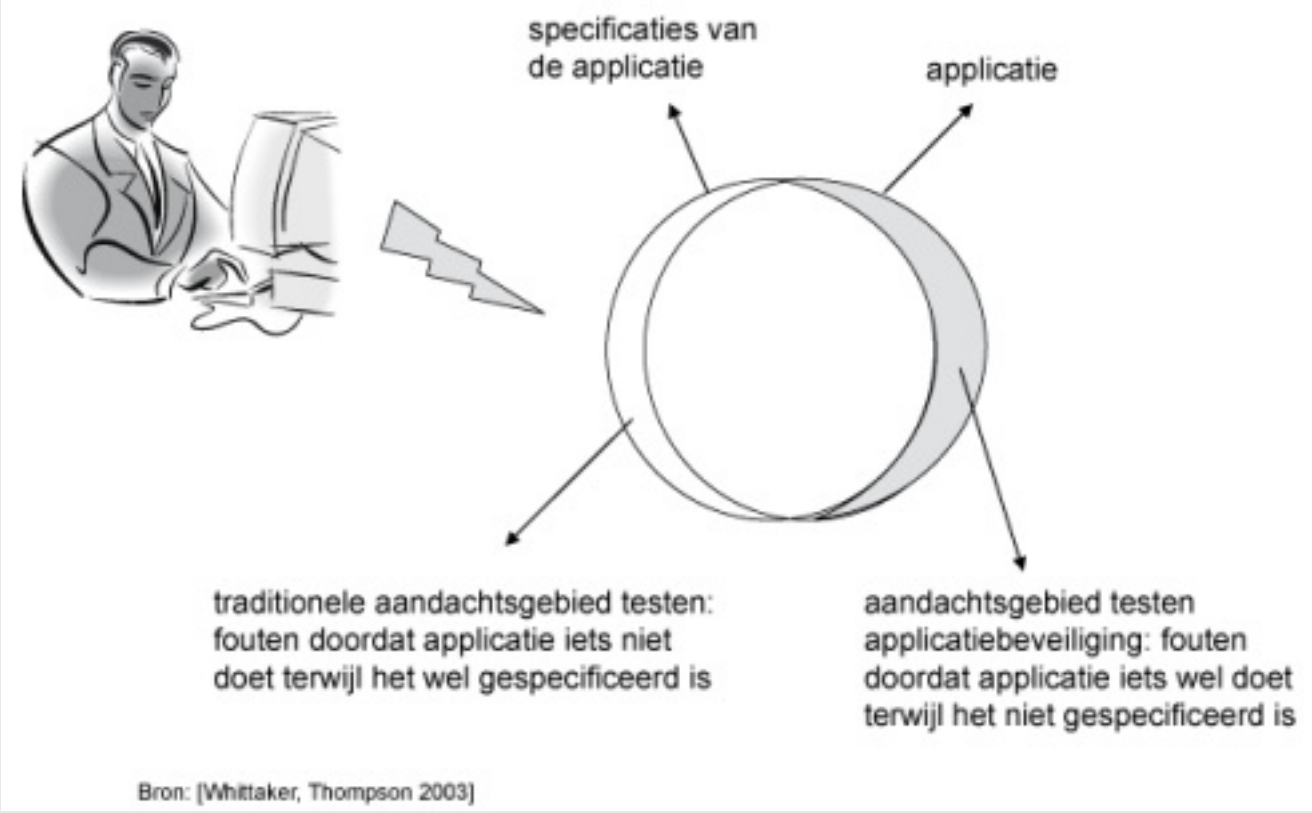
Hoewel achter de eerste vraag een wereld van beheersmaatregelen (onder andere toegangs- en autorisatiemanagement) schuilgaat, is de correctheid vrij gestandaardiseerd te testen, bijvoorbeeld met de semantische testtechniek.

De vraag: “zijn de ingevoerde gegevens juist” behoort al sinds jaar en dag tot het aandachtsgebied van functioneel testen en is met veel van de in hoofdstuk 14 beschreven testontwerptechnieken te testen.

De mogelijkheid informatie na invoering te manipuleren, anders dan via geautoriseerde/geoorloofde wijzigingen, behoort tot het aandachtsgebied van (op dit moment) gespecialiseerde bedrijven die hiertoe diepgaande security audits uitvoeren. Een bekende testvorm hierbij is het zogenaamde penetratietesten of ethical hacking. Hierbij wordt gebruik gemaakt van bekende zwakheden in de gebruikte software, van systeeminformatie die uit bijvoorbeeld cookies kan worden gehaald en van diverse soorten tools die ook “echte” hackers gebruiken. Er moet op worden gerekend dat de testhackers de beveiliging doorbreken. Zeker als dit externe inhuurkrachten zijn, moet dit van tevoren worden afgestemd met bijvoorbeeld de afdeling juridische zaken. De doorgebroken tester kan dan namelijk in de gelegenheid zijn om allerlei gevoelige informatie te bekijken of zijn doorbraak kan leiden tot onverwachte effecten waarvoor hij gevrijwaard wil zijn. Deze testers dienen uitgebreide technische kennis te hebben op de verschillende niveaus en dimensies van beveiliging.

Uitgediept

Eén van de lastigste aspecten bij het beveiligingstesten van een applicatie is dat de test zich erop moet richten of de applicatie dingen doet die *niet* gespecificeerd zijn. Bij de meer conventionele functionele tests is dit juist andersom: de tests worden gebaseerd op de specificaties en er wordt gecontroleerd of de applicatie werkt conform de specificaties. Zo mag vanuit beveiligingsoogpunt een applicatie bijvoorbeeld niet crashen bij verkeerde invoer en mogen (tussen)gegevens niet zichtbaar en manipuleerbaar zijn (tenzij gespecificeerd dat dit wel mag).



Figuur 10.1: Twee typen fouten.

Voor meer informatie over testen van beveiliging wordt verwezen naar [Splaine, 2002] en [Whittaker, 2003].

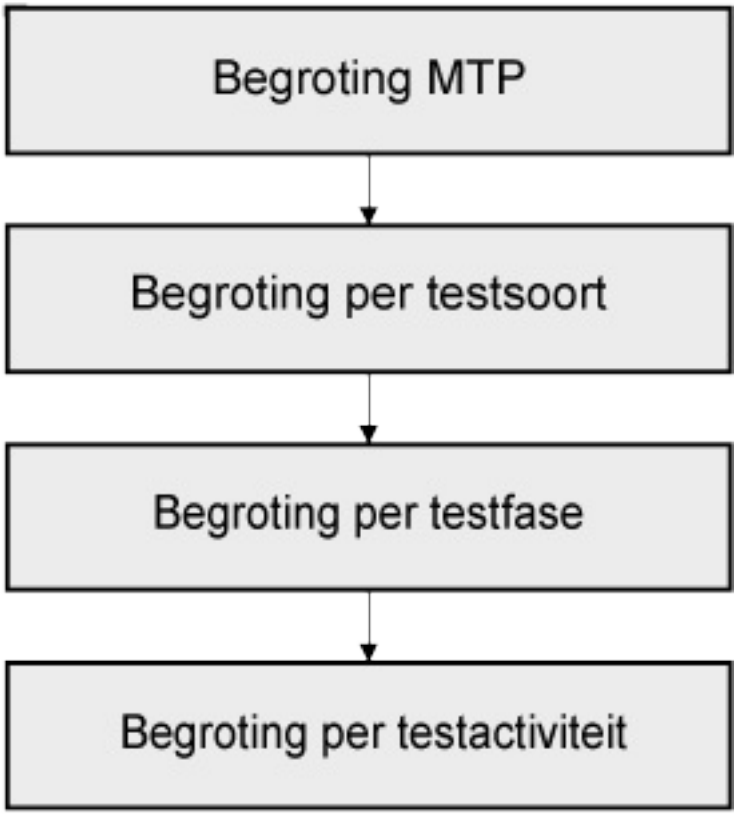
11 Begrotingstechnieken

Voor het opstellen van een begroting bestaan diverse technieken. Het hoofdstuk begint met een toelichting op de diverse niveaus waarop een begroting kan worden gemaakt en een overzicht van welke techniek voor het begroten van een bepaalde kwaliteitsattribuut geschikt is. Daarna worden de volgende begrotingstechnieken behandeld:

- Begroten op basis van verhoudingsgetallen
- Begroten op basis van testobject omvang
- Work Breakdown Structure
- Toetsbegrotingsaanpak
- Proportioneel begroten
- Extrapolatie
- Testpuntanalyse.

11.1 Begroten

Er zijn een aantal niveaus waarop een begroting kan worden gemaakt. De verschillende niveaus van begroting staan in figuur 11.1.



Figuur 11.1: Niveaus van begroting.

Het opstellen van een begroting voor een MTP gebeurt vroeg in het project. Veelal is dan nog niet alle kennis van het testobject aanwezig. Dit heeft als consequentie dat de nauwkeurigheid van de begroting beperkt is. De omvang en complexiteit van het testobject kan gedurende het project nog veranderen. Het is belangrijk dat de testmanager aan de belanghebbenden duidelijk maakt dat de begroting gebaseerd is op een aantal aannames en daarom later verder moet worden gedetailleerd. Een mogelijke oplossing hiervoor is om marges te gebruiken om de initiële begroting voor een MTP weer te geven.

De begroting in het MTP vormt het kader voor de begrotingen per testsoort (bijvoorbeeld systeemtest, gebruikersacceptatietest en productieacceptatietest). Binnen een testsoort wordt vervolgens de benodigde tijd voor de verschillende fasen, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding vastgesteld. Binnen de testfasen worden afzonderlijke testactiviteiten begroot. De tijd die nodig is voor het opstellen van het MTP (Planning) valt buiten de begrotingen. Hiervoor wordt vaak een vast aantal uren begroot. Het opstellen van het plan bestaat immers uit het uitvoeren van vastomlijnde activiteiten. Hierbij is de invloed van bijvoorbeeld de omvang van

het testobject, op de benodigde tijd voor het opstellen van het MTP, beperkt. Als er al invloed is, dan zal dit vooral merkbaar zijn bij de activiteiten “Analyseren productrisico’s” en “Bepalen teststrategie”. In de praktijk blijkt er aan het opstellen van het MTP tussen de 60 en 160 uur te worden besteed.

Naarmate de begroting later in het testtraject en dus op een lager niveau plaatsvindt, is meer kennis over het testobject beschikbaar. Bovendien kan dan gebruik worden gemaakt van ervaringen uit het begin van het traject. Daardoor wordt de begroting steeds nauwkeuriger.

Het opstellen van een begroting bestaat, onafhankelijk van het niveau, uit de volgende generieke stappen:

1. Inventariseer het beschikbare materiaal dat als basis voor het begroten kan dienen.
2. Kies (een aantal) begrotingstechnieken.
   Het verdient aanbeveling om meerdere technieken naast elkaar te gebruiken. De uitkomsten van de verschillende technieken kunnen zo met elkaar worden vergeleken. Naast begrotingstechnieken is het waardevol om een ervaren medewerker een inschatting van de benodigde tijd te vragen (expertbegroting).
3. Stel de uiteindelijke begroting vast.
   Doel van deze stap is om de uitkomsten uit de vorige stap bij elkaar te brengen tot één begroting. Als de uitkomsten dicht bij elkaar liggen, is middelen gerechtvaardigd. Anders moeten de verschillen worden geanalyseerd. Als ook na analyse van de verschillen geen goede begroting mogelijk is, moet overleg plaatsvinden met de opdrachtgever. De testmanager legt daarbij de problemen voor en doet voorstellen om te komen tot een goede begroting.
4. Presenteer de uitkomst.
   Doel van het presenteren van de uitkomst is om de consequenties van de gekozen teststrategie en aanpak voor de business inzichtelijk te maken. Het is hierbij belangrijk om duidelijk aan te geven welke aannames zijn gedaan. Met name bij een begroting die heel vroeg in het traject wordt opgesteld, zal sprake zijn van aannames die later in het traject concreter worden.

Met name het kiezen van de begrotingstechnieken is hierbij een stap waarvoor ervaring is vereist. Zoals aangegeven bestaan voor het opstellen van een begroting verschillende begrotingstechnieken. De verschillende begrotingstechnieken staan beschreven in de volgende paragrafen. Hierbij gelden de volgende uitgangspunten:

- Het begroten van de testactiviteiten in de ontwikkelfase (unittest en unitintegratietest) is een integraal deel van het begroten van het realisatieproject en valt buiten de beschouwing, behalve daar waar expliciet genoemd.
- Bij de vermelde technieken worden waar mogelijk ervaringscijfers genoemd. Bij deze cijfers wordt vermeld wat de achtergrond van de cijfers is. De getoonde cijfers dienen altijd te worden gezien in de beschreven context. Zij hoeven niet geldig te zijn in een andere situatie.
- Bij alle ervaringscijfers die in de komende paragrafen worden genoemd, is uitgegaan van één hertest.
- Voor het gestructureerd verzamelen en analyseren van testbegrotingscijfers wordt verwezen naar hoofdstuk 13 “Metrics”.

Om een goede keuze uit de verschillende technieken te maken, zijn twee hulpmiddelen beschikbaar. Deze hulpmiddelen geven antwoord op de volgende vragen:

- Welke techniek is geschikt voor welk niveau van begroting?
- Welke technieken zijn geschikt voor het begroten van welk(e) kwaliteitsattributen?

De antwoorden op deze vragen staan in de onderstaande tabellen.

	Mastertestplan	Detailtestplan	Testfase	Testactiviteit
Begroten o.b.v. verhoudingsgetallen	X	X	X	(X)
Begroten o.b.v. omvang	X			
Work breakdown structure (WBS)	X	X	X	X
Toetsbegrotingstechnieken	X			
Proportioneel begroten	X	X	X	(X)
Extrapolatie			X	X
Testpuntanalyse (begroten o.b.v. omvang en strategie)	X	X		

In de onderstaande tabel staan de mogelijke begrotingstechnieken per kwaliteitsattribuut. Hierbij is voor dynamische tests een onderscheid gemaakt in drie verschillende diepgangen, namelijk ● , ● ● en ● ● ● (laag, gemiddeld en hoog).

	Toets	Statische test	UT	UIT	Implementatie test	Dynamische test	Dynamische test	Dynamische test
Testzwaarte →	●	●	●	●		●		

							●●	●●●
Kwaliteitsattribuut ↓	Aantal blz 1)		2)	2)	3)			
Beheerbaarheid		Tpa-s4)				Wbs5)	-	
Beveiliging		Tpa-s				Tpa	Tpa	Wbs
Bruikbaarheid		Tpa-s				Tpa6)	Tpa6)	Wbs
Connectiviteit		Tpa-s				-	-	
Continuïteit		Tpa-s				Timebox7) week	Timebox7) maand	Timebox7) kwartaal
Controleerbaarheid		Tpa-s				-	-	
Flexibiliteit		Tpa-s				-	-	
Functionaliteit		Tpa-s	Urenbox7)	Urenbox7)		Tpa	Tpa	Tpa
Gebruikers-vriendelijkheid		Tpa-s				Wbs	Wbs	Wbs
Herbruikbaarheid		Tpa-s				-	-	
Infrastructuur		Tpa-s				-	-	
Inpasbaarheid		Tpa-s				Tpa6)	Tpa6)	Tpa
Onderhoudbaarheid		Tpa-s				-	-	
Performance Batch Online		Tpa-s				Tpa Wbs	Tpa Wbs	Tpa Wbs
Portabiliteit		Tpa-s				Wbs	Tpa	Tpa
Testbaarheid		Tpa-s				-	-	
Zuinigheid		Tpa-s				-	Wbs	

Toelichting bij de tabel:

- Het is niet mogelijk om voor deze diepgang een specifieke begrotingstechniek toe te passen.

1) Bij toetsen van een kwaliteitsattribuut moeten verschillende bladzijden worden gelezen. Kwaliteitsattributen die met functionaliteit te maken hebben vereisen bestudering van de bladzijden waar de functionaliteit wordt beschreven. Andere kwaliteitsattributen worden veelal in andere bladzijden beschreven. Dit leidt tot een verschillend aantal te toetsen bladzijden per kwaliteitsattribuut.
2) Uitgangspunt is dat de begroting van de standaard testactiviteiten in de UT en de UIT integraal onderdeel van de begroting van de realisatie is. Indien gewenst kan extra aandacht voor testen tijdens de UT en UIT worden aangegeven. De begrotingstechniek hiervoor is dan een urenbox, waarbij bijvoorbeeld een opslagpercentage bij de bouwinspanning (bijvoorbeeld 10%) of een deel van de inspanning voor de ST wordt geteld.
3) TPA-i is de component voor het impliciet testen van een kwaliteitsattribuut tijdens dynamisch testen van een ander kwaliteitsattribuut. In TPA geeft dat een additionele toevoeging van 0,02 bij het bepalen van de Q_d. Voor een uitgebreide omschrijving van TPA wordt verwezen naar paragraaf 11.8.
4) TPA-s is de statische component van TPA. Voor een uitgebreide omschrijving van TPA wordt verwezen naar paragraaf 11.8.
5) WBS = Work Breakdown Structure.
6) Indien bruikbaarheid en inpasbaarheid worden getest met dezelfde testvorm/testtechniek dan wordt de inspanning één keer meegeteld.
7) De timebox en urenbox worden bepaald door factoren buiten het testproces. Timebox week in de bovenstaande tabel betekent dat er een periode van één week wordt getest.

11.2 Begroten op basis van verhoudingsgetallen

Om verhoudingsgetallen als basis voor het opstellen van een begroting te kunnen gebruiken, is het belangrijk om zoveel mogelijk ervaringscijfers te verzamelen. Voor gelijksoortige projecten kunnen op die manier ‘standaard’ verhoudingsgetallen worden afgeleid. Gelijksoortige projecten zijn hierbij projecten die op belangrijke kenmerken met elkaar overeenkomen. Denk hierbij bijvoorbeeld aan dezelfde manier van ontwikkelen, hetzelfde ontwikkelplatform, dezelfde softwareomgeving, hetzelfde ervaringsniveau van de ontwikkelaars, enzovoort.

Vanzelfsprekend sluiten in het algemeen de verhoudingsgetallen van de eigen organisatie het best aan op de praktijk van die organisatie. Verhoudingsgetallen kunnen op alle niveaus van begroten worden gebruikt.

Op het niveau van testactiviteiten binnen testfasen zijn de verhoudingsgetallen echter zo specifiek dat deze alleen binnen de eigen organisatie en veelal zelfs alleen binnen het toepassingsgebied (project of systeem) kunnen worden gebruikt.

Hieronder staan een aantal verhoudingsgetallen tussen testen en andere ontwikkelactiviteiten uit de praktijk. Een

organisatie kan deze waarnemingen als startpunt gebruiken. Door vervolgens eigen ervaringscijfers bij te houden, kunnen de verhoudingsgetallen steeds beter op de eigen organisatie worden afgestemd.

Bij de verschillende waarnemingen is gebruik gemaakt van de volgende standaardisatie van begrippen:

- Functioneel ontwerp (FO) = functioneel detail ontwerp.
- Realisatie, bestaande uit technisch ontwerp (TO), programmeren (P), unit- en unitintegratietest (UT en UIT).
- Functionele test. De functionele test is de test van het kwaliteitsattribuut functionaliteit met als testbasis het FO. Het testen van dit kwaliteitsattribuut vindt hierbij plaats in de testsoorten ST en AT.

Waargenomen verhoudingen bij een gemiddeld risicoprofiel zijn:

- FO : Realisatie : Functionele test = 2 : 5 : 3
  In een omgeving met een formeel compleet FO, watervalontwikkelmethode, 3GL programmeertaal en een gestructureerde testaanpak. Voor de activiteiten in de onderhoudsfase bleken deze getallen ook te gelden, waarbij bij het testen alleen de test van de wijziging wordt uitgevoerd.
- (FO+TO) : (P + UT + UIT) : Functionele test = 1 : 3 : 3
  In een omgeving met een niet volledig uitgedetailleerd FO, ervaren ontwikkelaars die de FO's zelf laten vullen en een startende testaanpak.
- FO : Realisatie : Functionele test = 1 : 2 : 1,2
  In een omgeving met een formeel compleet FO, watervalontwikkelmethode, ervaren ontwikkelaars en een functionele test die in testdekking niet maximaal is, maar gestuurd wordt door risico en een gemaximaliseerd budget. De testaanpak is gestructureerd.

Binnen een testsoort kunnen verhoudingsgetallen worden gebruikt om de verschillende fasen te begroten. Ook hiervoor zijn waarnemingen uit de praktijk beschikbaar:

- Voor een systeemtest met goede maar complexe specificaties is de waargenomen verhouding als volgt: Voorbereiding 6%, Specificatie 54%, Uitvoering 21%, Afronding 2% en voor Beheer en Inrichting en beheer infrastructuur samen 17%.
- Voor een systeem met onvoldoende testbasis is de volgende verhouding waargenomen: Voorbereiding 21%, Specificatie 33%, Uitvoering 24%, Afronding 5% en voor Beheer en Inrichting en beheer infrastructuur samen 17%.

Noot: in beide gevallen werd er 160 uur aan het opstellen van het MTP besteed.

11.3 Begroten op basis van testobjectomvang

De omvang van het testobject kan op een aantal manieren worden vastgesteld. Hier wordt de term Testobject Omvang Meter (TOM) gebruikt om op een eenduidige manier de omvang van een testobject aan te geven.

Op basis van een op deze manier vastgestelde testobject omvang kan voor de schatting van de functionele test, ook zonder dat de strategie (al) bekend is, het volgende kengetal worden gehanteerd:

1,5 tot 4 uur per omvangseenheid (TOM)

Het daadwerkelijke kengetal voor een specifiek toepassingsgebied is afhankelijk van:

- type omgeving (web, database)
- geboden ondersteuning
- kwaliteit testbasis
- omvang project, bij een heel klein en heel groot project richting factor 2
- vereiste rapportage
- ervaring testers.

Om een steeds betrouwbaardere inschatting te kunnen maken, kunnen organisaties ervaringscijfers bijhouden.

De omvang van een testobject (en daarmee dus het aantal TOM) kan op de volgende manieren worden vastgesteld:

- Gedetailleerde functionele beschrijving
  Op een gedetailleerde functionele beschrijving (bijvoorbeeld een functioneel ontwerp) kan een functiepuntanalyse

worden uitgevoerd. Het resultaat van de functiepuntanalyse is een aantal functiepunten (FP). Eén functiepunt wordt vervolgens gelijk gesteld aan één TOM, waardoor de omvang van het testobject (= aantal TOM) gelijk is aan het aantal functiepunten.

- Gegevensmodel
  Als er een gegevensmodel beschikbaar is, kan de volgende aanpak worden gebruikt om de omvang van het testobject te bepalen: bepaal het aantal logische gegevensverzamelingen (LGV'en) en schat de complexiteit. Door het aantal gegevensverzamelingen te vermenigvuldigen met de waarde uit de onderstaande tabel ontstaat de omvang van het testobject.

Aantal LGV'en	Complexiteit		
	Eenvoudig	Gemiddeld	Moeilijk
< 10	25	28	35
10 - 25	28	35	42
> 25	35	42	47

- Bladzijden requirements
  Binnen de literatuur zijn ervaringscijfers beschikbaar om de omvang van het testobject te relateren aan het aantal bladzijden requirements. In het algemeen is dan niet alle informatie beschikbaar over de omstandigheden waarin de gegevens zijn gemeten.
  – 1 A4-pagina requirements zonder diagrammen = 15 TOM [Collard, 1999].
  – In een groot klassiek traject, waarin een zeer gedetailleerd functioneel ontwerp zonder plaatjes voorhanden was, is het volgende ervaringscijfer gemeten:
  1 A4-pagina requirements = 2,5 à 3 TOM.
- Aantal schermen
  Als het aantal schermen bepalend is voor de omvang van de applicatie kan de volgende afleiding worden gebruikt [Collard, 1999]:
  1 scherm (window/webpage) = 8 TOM.
- Programmabroncode
  Voor een nieuwbouwproject is de programmabroncode natuurlijk pas beschikbaar na het realisatietraject. Voor bijvoorbeeld een migratietraject of een onderhoudstraject kunnen de onderstaande afleidingen wel bruikbaar zijn:
  – 1 kilo lines of code (3 GL) = 17 TOM [Collard, 1999].
  – [Jones, 1996]

1 KLOC (kilo lines of code)	Aantal TOM
C	6,6
Algol, Cobol, Fortran	10
PL/1	12
Lisp, basic	16
4GL database	25
Objective C	39
Smalltalk	49
Query talen	60

11.4 Work Breakdown Structure

Work Breakdown Structure (WBS) is een begrotingsaanpak die gebaseerd is op het opdelen van de werkzaamheden in deelactiviteiten tot een detailniveau waarop de benodigde tijd per activiteit kan worden geschat. Door de benodigde tijd voor de deelactiviteiten bij elkaar op te tellen, ontstaat de totaal benodigde tijd.

In de onderstaande tabel is de hoeveelheid uren per kwaliteitsattribuut aangegeven. Bij kwaliteitsattributen waarbij de strategie van belang is, is dit aangegeven. De uren komen uit de praktijk. Wel is de ervaringbase en daarmee de hardheid van de cijfers verschillend. Hardheden zijn:

- Hard: ervaring uit meer projecten; is vanuit verschillende bronnen bevestigd
- Ervaring: is op enkele bronnen gebaseerd
- Zacht: een inschatting door ervaren testconsultants.

In de praktijk blijkt welke factoren de uiteindelijke hoeveelheid uren het meest beïnvloeden. Deze factoren zijn weergegeven.

Kwaliteitsattribuut	Strategie	Schatting hardheid	Uren	Belangrijke factoren voor de variatie in grootte
Beheerbaarheid		zacht	24	

Installeerbaarheid				
Beveiliging	● ● ●	ervaring	80	Minimaal, urenbox
Bruikbaarheid	● ● ●	zacht	350	Incl. uren van de gebruikers
Continuïteit	● ● ●	nvt		Hangt af van de duur van schaduwproductie
Gebruikersvriendelijkheid	●	hard	70	Omvang applicatie (grens 15 / 100 schermen) Onderzoeksvraag grootte (grens: enkele onderwerpen)
Gebruikersvriendelijkheid	● ●	hard	80	
Gebruikersvriendelijkheid	● ● ●	hard	130	
Performance, Online	● ●	hard	192 tot 224	Laag: 15 gebruikerstaken Hoog: 40 gebruikerstaken Complexe database
Portabiliteit	●	zacht	28	
Zuinigheid	●	zacht	28	

N.b.: In de bovenstaande tabel zijn geen uren meegenomen voor bijvoorbeeld de inrichting van een bruikbaarheidslab of het selecteren van testondersteunende pakketten. Uitgangspunt is dat de benodigde faciliteiten beschikbaar zijn.

11.5 Toetsbegrotingsaanpak

Een in de literatuur veel genoemde omvangsbasis voor het uitvoeren van toetsen is het aantal bladzijden van het document dat wordt getoetst.

Cijfers uit de literatuur: 1-4 pagina's per uur per toetser per omvangseenheid, afhankelijk van:

- het aantal kwaliteitsattributen waarnaar wordt gekeken
- de ervaring van de toetser
- de gewenste diepgang
- de formaliteit van de toetsvorm; hoe formeler de toets hoe meer tijd de toetsing kost.

11.6 Proportioneel begroten

Deze begrotingstechniek gaat uit van een totale hoeveelheid budget voor het testen van het gehele testobject. Dit totaal wordt over de onderkende onderdelen verdeeld. Bij het verdelen van het totale budget over de verschillende onderdelen wordt rekening gehouden met de toegekende risicoklasse (bij de teststrategie) en de omvang van de onderdelen. Per risicoklasse (uit de teststrategie) wordt hierbij een factor gekozen die een gewogen verdeling mogelijk maakt. Bijvoorbeeld:

- Risicoklasse A krijgt een factor 1,5
- Risicoklasse B krijgt een factor 1
- Risicoklasse C krijgt een factor 0,6.

De stappen om vervolgens te komen tot een begroting zijn:

1. Bereken het product van de omvang van het te testen deelobject met de factor die hoort bij de risicoklasse van dat deelobject.
2. Tel de uitkomsten uit stap 1 bij elkaar op.
3. Bepaal de schalingsfactor door het aantal te verdelen uren te delen door het resultaat van stap 2.
4. Bereken het aantal uren per deelobject door de resultaten van stap 1 te vermenigvuldigen met de schalingsfactor.

Een voorbeeld om dit te verduidelijken:
Uitgangspunt is het verdelen van 100 uur over 5 deelobjecten (deelobject 1 t/m deelobject 5). Voor ieder deelobject is de omvang en een risicoklasse bepaald. Daarna wordt volgens de bovenstaande stappen het aantal uren per deelobject vastgesteld.

Deelobject	Omvang	Risicoklasse	Factor	Omvang x Factor	Schalingsfactor	Aantal uren
1	10	C	0,6	6		7,86
2	15	A	1,5	22,5		29,48
3	7	B	1	7		9,17
4	25	A	1,5	37,5		49,12
5	5	C	0,6	3		3,93
Totaal				**76**	**100/76=1,31**	**100 (100,56)**

11.7 Extrapolatie

Bij deze methode van begroting worden zo vroeg mogelijk in het traject metingen gedaan waardoor ervaringscijfers kunnen worden opgebouwd. Als bekend is welk percentage van de werkzaamheden in hoeveel tijd is afgerond, kan (bij benadering) worden vastgesteld hoeveel tijd nog nodig is voor de rest van de activiteiten. Deze methode wordt in de praktijk vooral veel gebruikt bij het begroten van testactiviteiten binnen een testsoort. Ook is deze methode erg geschikt voor het begroten van testactiviteiten binnen incrementele ontwikkelingsmethoden.

11.8 Testpuntanalyse

In deze paragraaf wordt de testbegrotingstechniek testpuntanalyse (TPA) beschreven. Met behulp van testpuntanalyse kan op een objectieve wijze een begroting worden gemaakt voor een systeemtest of een acceptatietest. Ontwikkeltesten is een impliciet onderdeel van de ontwikkelbegroting en valt daardoor buiten de scope van TPA. Om TPA toe te kunnen passen moet de omvang van het informatiesysteem bekend zijn. Hiervoor worden de resultaten van een functiepuntanalyse (FPA) gebruikt. FPA is een methode met de mogelijkheid om een technologieonafhankelijke meting te doen van de omvang van de door een geautomatiseerd systeem geboden functionaliteit en deze meting te gebruiken als basis voor productiviteitsmeting, het schatten van de benodigde middelen en projectbeheersing. De productiviteitsfactor omvat bij functiepuntanalyse wél de ontwikkeltesten, maar niet de acceptatie- en systeemtesten.

Testpuntanalyse kan tevens worden toegepast indien het aantal te besteden testuren van te voren reeds is vastgesteld. Door het uitvoeren van een testpuntanalyse kunnen de eventuele risico's die worden gelopen duidelijk worden aangetoond door de objectieve testpuntanalysebegroting te vergelijken met het van te voren vastgestelde aantal testuren. Tevens is het mogelijk met behulp van testpuntanalyse het relatieve belang tussen de diverse functies te bepalen c.q. te berekenen, op basis waarvan de beschikbare testtijd zo optimaal mogelijk kan worden besteed.

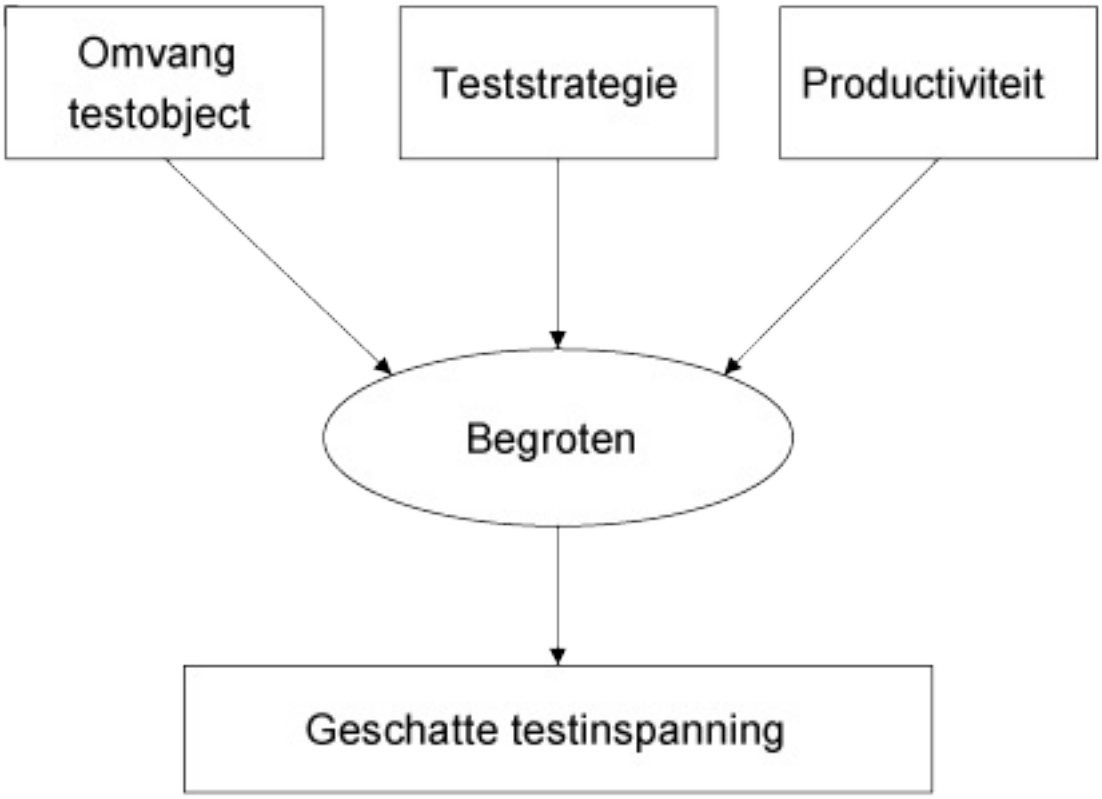
Ook kan testpuntanalyse worden toegepast om in een vroegtijdig stadium een globale testbegroting te maken (paragraaf 11.8.8).

Filosofie

Bij het bepalen van een testbegroting in het kader van een acceptatie- of systeemtest speelt een drietal elementen een rol (zie figuur 11.2 "Basiselementen begroten"):

- De omvang van het informatiesysteem dat moet worden getest.
- De teststrategie (welke deelobjecten en welke kwaliteitsattributen moeten worden getest en met welke diepgang?).
- De productiviteit.

De eerste twee elementen bepalen samen de omvang van de uit te voeren test (uitgedrukt in testpunten). Indien het aantal testpunten wordt vermenigvuldigd met de productiviteit (de hoeveelheid tijd benodigd om een bepaalde testomvang uit te voeren), resulteert dit in een testbegroting in uren. De drie elementen zijn hierna nader uitgewerkt.



Figuur 11.2: Basiselementen begroten.

Omvang

Met omvang wordt in dit kader de omvang van het informatiesysteem bedoeld. De omvang van een informatiesysteem is binnen testpuntanalyse met name gebaseerd op het aantal functiepunten. Op de functiepuntanalyse dient in het kader van de testpuntanalyse echter een aantal toevoegingen c.q. aanpassingen te worden uitgevoerd. Bij het testen is namelijk een aantal factoren te onderkennen die bij het bepalen van het aantal functiepunten geen of nauwelijks een rol spelen, maar bij testen van essentieel belang zijn. Deze factoren zijn:

- Complexiteit
  Hoeveel condities zijn er aanwezig in een functie. Meer condities betekent vrijwel automatisch meer testgevallen en dus meer testinspanning.
- Systeemdoorwerking
  Hoeveel gegevensverzamelingen worden door de functie onderhouden en hoeveel andere functies maken van deze gegevensverzamelingen gebruik. De ‘andere’ functies dienen tevens te worden getest indien deze onderhoudsfunctie wordt aangepast.
- Uniformiteit
  Is de structuur van een functie van dien aard dat reeds bestaande testspecificaties met slechts een geringe aanpassing hergebruikt kunnen worden. Met andere woorden bestaan binnen het informatiesysteem meerdere functies met dezelfde structuur?

Teststrategie

Tijdens de systeemontwikkeling of het onderhoud worden kwaliteitseisen gespecificeerd ten aanzien van het informatiesysteem. Tijdens het testen moet worden vastgesteld in hoeverre aan de gestelde kwaliteitseisen is voldaan. Er is echter nooit sprake van een onbeperkte hoeveelheid testmiddelen en testtijd. Daarom is het belangrijk om de testinspanning te relateren aan de verwachte productrisico’s. Met behulp van een productrisicoanalyse (hoofdstuk 9) worden onder andere testdoelen, relevante kenmerken per testdoel, te onderscheiden deelobjecten per kenmerk en de risicoklasse per kenmerk/ deelobject vastgesteld. Vervolgens wordt het resultaat van de productrisicoanalyse gebruikt om de teststrategie op te stellen. Hierbij zal een combinatie van een kenmerk/deelobject uit een hoge risicoklasse bij de vertaling naar de teststrategie vaak om diepgaande, zware tests en daardoor relatief veel testinspanning vragen. De teststrategie vormt een input voor de testpuntanalyse. Binnen testpuntanalyse vindt een vertaling van de teststrategie in de benodigde testtijd plaats.

Naast de algemene kwaliteitseisen van het informatiesysteem zijn er verschillen met betrekking tot de kwaliteitseisen

tussen de functies onderling. Het betrouwbaar functioneren van sommige functies is van essentieel belang voor het bedrijfsproces. Deze functies zijn de reden dat het informatiesysteem is ontwikkeld. Vanuit gebruikersoptiek is de functie die de hele dag intensief wordt gebruikt wellicht veel belangrijker dan de verwerkingsfunctie die 's nachts draait. Per functie is er derhalve een tweetal (subjectieve) factoren dat de mate van diepgang bepaalt: het gebruikersbelang van de functie en de gebruiksintensiteit. De diepgang geeft als het ware de mate van zekerheid oftewel inzicht in de kwaliteit weer, die de opdrachtgever wenst. De factoren gebruikersbelang en gebruiksintensiteit zijn uiteraard gebaseerd op de teststrategie.

De teststrategie geeft aan welke combinaties van kenmerk/deelobject met welke diepgang moeten worden getest. Hierbij wordt als kenmerk vaak een kwaliteitsattribuut gekozen. Binnen de testpuntanalyse wordt eveneens gemaakt van kwaliteitsattributen, waardoor dit nauw verbonden is met de teststrategie en in de praktijk veelal grotendeels gelijktijdig wordt uitgevoerd.

Tip

Koppelen TPA parameters aan teststrategie risicoklassen

TPA kent vele parameters die het benodigd aantal uren bepalen. De risicoklassen uit de teststrategie zijn goed te vertalen naar deze parameters. De TPA parameters kennen in het algemeen drie waarden, die vervolgens goed aan de drie risicoklassen uit de teststrategie kunnen worden gekoppeld (risicoklasse A, B en C).

Als er géén gedetailleerde informatie beschikbaar is om het testobject te verdelen in de verschillende risicoklassen kan de volgende verdeling worden gebruikt:

- 25% risicoklasse A
- 50% risicoklasse B
- 25% risicoklasse C.

Deze verdeling dient vervolgens als uitgangspunt voor het uitvoeren van een TPA.

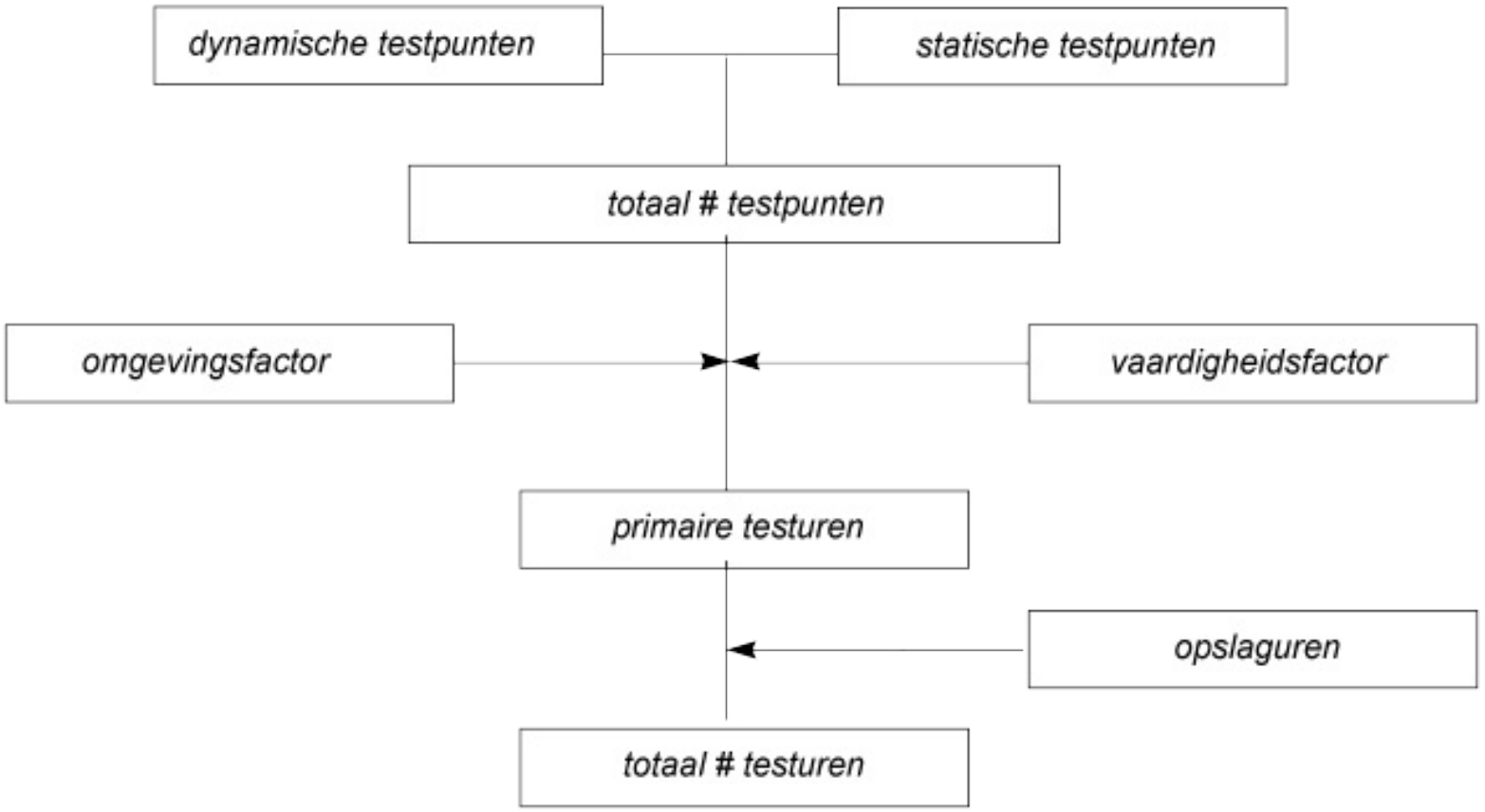
Productiviteit

De toepassing van dit begrip is niet nieuw voor iemand die reeds eerder begrotingen heeft gemaakt op basis van functiepunten. De productiviteit legt bij functiepuntanalyse de relatie tussen inspanningsuren en het gemeten aantal functiepunten. Voor testpuntanalyse betekent productiviteit de tijd die nodig is om één testpunt, bepaald door de omvang van het informatiesysteem en de teststrategie, te realiseren. De productiviteit is opgebouwd uit een tweetal onderdelen: de vaardigheidsfactor en de omgevingsfactor. De vaardigheidsfactor is met name gebaseerd op de kennis en kunde van het testteam. Dit cijfer is daardoor organisatie en zelfs persoon specifiek. De omgevingsfactor geeft aan in welke mate de omgeving van invloed is op de testactiviteiten waaraan de productiviteit is gerelateerd. Het gaat hierbij om aspecten zoals de beschikbaarheid van testtools, de ervaring met de betreffende testomgeving, de kwaliteit van de testbasis en de eventuele beschikbaarheid van testware.

Globale werking

De werking van testpuntanalyse is schematisch op de volgende pagina weergegeven.

Op basis van het aantal functiepunten per functie, de functie afhankelijke factoren (complexiteit, doorwerking, uniformiteit, gebruikersbelang en gebruiksintensiteit) en de kwaliteitseisen c.q. teststrategie met betrekking tot de dynamisch te meten kwaliteitsattributen wordt per functie het aantal testpunten berekend dat nodig is om de dynamisch meetbare kwaliteitsattributen te testen (dynamisch meetbaar betekent dat een oordeel kan worden gevormd over een bepaald kwaliteitsattribuut door het uitvoeren van programma's). Sommering van deze testpunten over de functies geeft het aantal dynamische testpunten.



Figuur 11.3 : Schematische weergave testpuntanalyse.

Op basis van het totale aantal functiepunten van het informatiesysteem en de kwaliteitseisen c.q. teststrategie met betrekking tot de statische kwaliteitsattributen, wordt het aantal testpunten bepaald dat nodig is om de statisch meetbare kwaliteitsattributen te testen (statisch testen: testen door het controleren en onderzoeken van producten, zonder dat er sprake is van het uitvoeren van programma's). Dit levert het aantal statische testpunten. Het totaal aantal testpunten ontstaat door de dynamische en de statische testpunten op te tellen.

De primaire testuren worden vervolgens verkregen door het totaal aantal testpunten te vermenigvuldigen met de berekende omgevingsfactor en de geldende vaardigheidsfactor. Het aantal primaire testuren geeft de tijd aan die nodig is om de primaire testactiviteiten te kunnen uitvoeren. Met andere woorden de tijd die nodig is om de testactiviteiten van de fasen Voorbereiding, Specificatie, Uitvoering en Afronding van het TMap-faseringmodel uit te voeren.

Het aantal uren (opslaguren), dat nodig is voor de uitvoering van secundaire testactiviteiten uit de fase Beheer en uit de fase Inrichting en beheer infrastructuur, wordt berekend als een percentage van de primaire testuren.

Tot slot wordt het totaal aantal testuren verkregen door het primaire aantal testuren te vermeerderen met het aantal opslaguren. Het totaal aantal testuren is een begroting voor alle TMap-testactiviteiten behalve het opstellen van het testplan.

Uitgangspunten

Met betrekking tot testpuntanalyse gelden de volgende uitgangspunten:

- Testpuntanalyse beperkt zich tot de kwaliteitsattributen die 'meetbaar' zijn. Meetbaar zijn betekent dat voor het betreffende kwaliteitsattribuut een testtechniek beschikbaar is. Tevens dient met betrekking tot deze testtechniek in relatie tot het betreffende kwaliteitsattribuut voldoende praktijkervaring te zijn opgedaan om concrete uitspraken te kunnen doen over de benodigde testinspanning.
- In relatie tot het voorafgaande uitgangspunt kan worden vastgesteld dat niet alle mogelijke kwaliteitsattributen die kunnen voorkomen, zijn opgenomen in de huidige versie van testpuntanalyse. Dit heeft als reden dat er (nog) geen concrete testtechniek beschikbaar is of dat met een testtechniek nog te weinig praktijkervaring is opgedaan en er derhalve nog geen voldoende betrouwbare metrics voorhanden zijn. Een eventuele volgende versie van testpuntanalyse omvat wellicht meer kwaliteitsattributen.
- Testpuntanalyse is in beginsel persoonsonafhankelijk. Met andere woorden verschillende personen die een testpuntanalyse uitvoeren op hetzelfde informatiesysteem moeten in principe tot dezelfde begroting komen. Een en

ander wordt bereikt door alle factoren die niet objectief te classificeren zijn door de opdrachtgever te laten bepalen en voor alle factoren waarvoor dat wel mogelijk is een eenduidige klasse-indeling te hanteren.
- Testpuntanalyse kan worden uitgevoerd als er een functiepunttelling conform NESMA [NESMA, 1996] of IFPUG [IFPUG, 1994] beschikbaar is; hierbij worden de bruto functiepunten als uitgangspunt genomen.
- Materiekennis wordt binnen testpuntanalyse niet gezien als een factor die van invloed is op benodigde hoeveelheid testinspanning. Wel is het uiteraard belangrijk een bepaalde hoeveelheid materiekennis binnen het testteam aanwezig te hebben. Materiekennis is in deze optiek een randvoorwaarde die dient te worden ingevuld tijdens het opstellen van het testplan.
- Binnen testpuntanalyse wordt bij het bepalen van de begroting uitgegaan van gemiddeld één complete hertest. Dit gemiddelde is een gewogen gemiddelde op basis van omvang van de functies, uitgedrukt in testpunten.

Tip

Van COSMIC full function points (CFFP) naar function points (FP)

Voor het begroten van de projectomvang wordt naast de Function Point Analysis (FPA) aanpak steeds vaker de COSMIC[1] Full Function Points (CFFP) aanpak gebruikt [Abran, 2003]. FPA is ontstaan in een periode waar alleen sprake was van een mainframe omgeving, daarbij leunt FPA zwaar op het verband tussen tussen functionaliteit en het gegevensmodel. CFFP houdt echter ook rekening met andere architecturen als ‘client server’ en ‘multi tier’ én ontwikkelmethoden als ‘objected oriënted’, ‘component based’ en RAD. Voor het omrekenen van CFFP naar function points (FP) kan de volgende vuistregel worden gebruikt:

- bij CFFP < 250 : FP = CFFP
- bij 250 ≤ CFFP ≤ 1000 : FP = CFFP / 1,2
- bij CFFP > 1000 : FP = CFFP / 1,5

TPA, de techniek in detail

11.8.1 Invoer en startvoorwaarden

Om een testpuntanalyse uit te kunnen voeren, dient men te beschikken over een functioneel ontwerp. Dit functioneel ontwerp dient gedetailleerde procesbeschrijvingen evenals een logisch datamodel, bij voorkeur inclusief een CRUD-matrix, te omvatten.

Tevens dient er een functiepunttelling te zijn uitgevoerd conform NESMA of IFPUG. Deze functiepuntmethoden zijn bruikbaar als invoer voor TPA. Het is van belang dat bij het bepalen van de vaardigheidsfactor, op basis van ervaringscijfers, slechts een van deze functiepuntmethoden wordt gehanteerd en niet meerdere door elkaar. Bij een functiepunttelling wordt het aantal bruto functiepunten als uitgangspunt genomen. De gehanteerde functiepuntmethode is niet van belang bij het bepalen van de testpunten. Op de vaardigheidsfactor kan dit wel invloed hebben.

Op de functiepunttelling dienen in het kader van TPA de volgende aanpassingen te worden aangebracht:

- De functiepunten van de (logische) gegevensverzamelingen die worden onderkend binnen de functiepunttelling dienen te worden toegekend aan de functie(s) die de invoer van de betreffende (logische) verzameling verzorgt.
- De functiepunten van de koppelingsgegevensverzamelingen die worden onderkend binnen de functiepunttelling worden toegekend aan de functie (of eventueel functies) die gebruik maken van de betreffende koppelingsgegevensverzameling.
- Voor FPA-functies van de klasse kloon wordt het aantal functiepunten gehanteerd dat geldt voor de originele FPA-functie. Een kloon is een FPA-functie die reeds is gespecificeerd en/of gerealiseerd binnen een andere of dezelfde gebruikersfunctie binnen het project.
- Voor FPA-functies van de klasse dummy wordt zo mogelijk het aantal functiepunten bepaald, anders krijgt deze FPA-functie de kwalificatie gemiddelde complexiteit en het corresponderende aantal functiepunten. Een dummy is een FPA-functie als de functionaliteit niet behoeft te worden gespecificeerd en/of gerealiseerd, maar beschikbaar is omdat dit reeds is gebeurd buiten het project.

Tip

Schattingsrichtlijn voor het tellen van functiepunten

Indien er geen functiepunttelling aanwezig is en men wenst deze alsnog (ten behoeve van TPA) uit te voeren, kan voor het bepalen van de benodigde tijd om de functiepunten te tellen de volgende richtlijn worden gehanteerd: Bepaal het aantal TOM volgens één van de in paragraaf 11.3 genoemde manieren en deel dit door 400. De uitkomst geeft een schatting van het aantal dagen dat nodig is om de functiepunten te tellen.

Noot: als regel geldt dat het mogelijk is om per dag 350 à 400 functiepunten te tellen.

Rekenvoorbeeld (1): Aantal functiepunten (FPf)

Een informatiesysteem heeft twee gebruikersfuncties en één interne logische gegevensverzameling:

Registreren (11 functiepunten) met als onderliggende FPA-functies:

Invoeren	3 functiepunten
Wijzigen	4 functiepunten
Verwijderen	4 functiepunten

Verwerken (12 functiepunten) met als onderliggende FPA-functies:

Overzicht 1	5 functiepunten
Overzicht 2	7 functiepunten

De interne logische gegevensverzameling ‘gegevens’ heeft 7 functiepunten en wordt in het kader van testpuntanalyse toegerekend aan de invoerfunctie.

FP_f Registreren	18 functiepunten
FP_f Verwerken	12 functiepunten

(FP_f = functiepunten per functie)

11.8.2 Dynamische testpunten

Het aantal dynamische testpunten is een optelling van het aantal testpunten per functie met betrekking tot dynamisch meetbare kwaliteitsattributen. Voor het berekenen van het aantal testpunten per functie zijn de factoren in twee categorieën ingedeeld:

- afhankelijk van de functie (A_f)
- de kwaliteitseisen met betrekking tot de dynamisch te testen kwaliteitsattributen (Q_d).

Als eenheid van functie wordt de FPA-functie gehanteerd. Bij het bepalen van het gebruikersbelang en gebruiksintensiteit staat de gebruikersfunctie als communicatiemiddel centraal. Het belang dat door de gebruikers wordt toegekend aan de gebruikersfunctie geldt tevens voor alle onderliggende FPA-functies.

Functie-afhankelijke factoren

Hieronder worden de functie-afhankelijke factoren beschreven, inclusief de bijbehorende waarderingen. Het is slechts mogelijk voor een van de drie beschreven waarden te kiezen (tussenwaarden zijn derhalve niet toegestaan). Indien er te weinig informatie beschikbaar is om een bepaalde factor te classificeren dan dient deze de nominale waarde (in deze paragraaf vet gedrukt) te krijgen.

Gebruikersbelang

Gebruikersbelang is gedefinieerd als het relatieve belang dat de gebruiker aan een bepaalde functie hecht in relatie tot de overige binnen het systeem aanwezige functies. Als vuistregel geldt dat ongeveer 25% van de functies in de categorie “hoog” terecht dienen te komen, 50% in de categorie “neutraal” en 25% in de categorie “laag”.

Gebruikersbelang wordt toegekend aan de door de gebruiker benoemde functionaliteit. Dit betekent toekenning van het gebruikersbelang aan de gebruikersfunctie. Het bepalen van het gebruikersbelang van een functie dient uiteraard in overleg met de opdrachtgever en andere vertegenwoordigers van de gebruikersorganisatie te geschieden.

Weging

3 laag: het relatieve belang van de betreffende functie in relatie tot de overige functies is laag
6 neutraal: het relatieve belang van de betreffende functie is neutraal in relatie tot de overige functies
12 hoog: het relatieve belang van de betreffende functie in relatie tot de overige functies is hoog.

Gebruiksintensiteit

Gebruiksintensiteit is gedefinieerd als de frequentie waarmee een bepaalde functie wordt gebruikt door de gebruiker en de omvang van de gebruikersgroep die de betreffende functie gebruikt.

Evenals gebruikersbelang wordt gebruiksintensiteit toegekend aan door gebruikers benoemde functionaliteit c.q. de gebruikersfunctie.

Weging

2 laag: de functie wordt slecht enkele malen per dag of per week door de gebruikersorganisatie uitgevoerd
4 neutraal: de functie wordt een groot aantal keren per dag door de gebruikersorganisatie uitgevoerd
8 hoog: de functie wordt continu (minimaal 8 uur per dag) uitgevoerd.

Systeemdoorwerking

Systeemdoorwerking is de mate waarin een mutatie die plaatsvindt binnen de betreffende functie doorwerkt in het systeem. De mate van doorwerking wordt bepaald door eerst de logische gegevensverzamelingen (LGV’en) waarop de betreffende functie een mutatie kan uitvoeren te bepalen en vervolgens het aantal andere functies (binnen de systeemgrenzen) te bepalen die de betreffende LGV benaderen.

De waardering van de doorwerking vindt vervolgens plaats met behulp van een matrix waarin verticaal het aantal LGV’en is aangegeven dat wordt gemuteerd door de functie en horizontaal het aantal andere functies dat de betreffende LGV’en benadert. Een bepaalde functie kan eventueel meerdere malen worden meegeteld in het kader van de doorwerking, dit omdat de betreffende functie meerdere LGV’en benadert die alle door de onderhavige functie worden onderhouden.

Aantal LGV’en	**Functies**		
	1	**2-5**	**> 5**
1	L	L	M
2 - 5	L	M	H
> 5	M	H	H

Toelichting: L = Lage doorwerking, G = Gemiddelde doorwerking, H = Hoge doorwerking.

Indien een functie geen LGV muteert valt deze functie in de categorie lage doorwerking. Bij het bepalen van de systeemdoorwerking is een CRUD-matrix uitermate behulpzaam.

Weging

2 de functie heeft een lage doorwerking
4 de functie heeft een gemiddelde doorwerking
8 de functie heeft een hoge doorwerking.

Complexiteit

De waardering van de complexiteit van een functie vindt plaats op basis van een uitspraak over het algoritme. De globale opzet van het algoritme kan zijn beschreven middels pseudo-code, Nassi-Shneidermann of gewone tekst. De mate van complexiteit van de functie wordt bepaald door het aantal condities binnen het algoritme van de betreffende functie vast te stellen. Bij het tellen van het aantal condities dient slechts het verwerkingsalgoritme in beschouwing te worden genomen. Condities die het gevolg zijn van database controles, zoals validaties op domein of controle op fysieke aanwezigheid worden niet meegenomen aangezien deze reeds impliciet in de functiepunttelling zijn meegenomen.

De complexiteit kan dus eenvoudig worden bepaald door het aantal condities te tellen. Samengestelde condities zoals IF a AND b THEN, dragen dubbel bij aan de complexiteit. Immers zonder het AND-statement zouden er twee IFstatements voor nodig zijn. Zo wordt een CASE-statement met n cases geteld voor n-1 condities, omdat de vervanging van het CASE-statement door achtereenvolgende IF-statements, n-1 condities zou opleveren. Samengevat: tel de condities, niet de operatoren.

Weging

3	binnen de functie zijn maximaal 5 condities aanwezig
6	binnen de functie zijn minimaal 6 en maximaal 11 condities aanwezig
12	binnen de functie zijn meer dan 11 condities aanwezig.

Uniformiteit

In een drietal situaties hoeft een functie slechts voor 60% te worden meegeteld:

- Bij een tweede voorkomen van een bijna unieke functie dient de tweede functie slechts voor 60% te worden meegeteld. In dit geval kunnen de op te stellen testspecificaties namelijk grotendeels worden hergebruikt.
- Klonen dienen slechts voor 60% te worden meegeteld. Ook in dit geval kunnen de op te stellen testspecificaties worden hergebruikt.
- Dummy functies dienen slechts voor 60% te worden meegeteld. Dit geldt slechts indien er voor de betreffende dummy reeds testspecificaties zijn opgesteld en geschikt zijn voor hergebruik.

De factor uniformiteit krijgt de waarde 0,6 als aan één van de bovenstaande voorwaarden wordt voldaan de en anders de waarde 1.

Er kunnen dus binnen een informatiesysteem functies bestaan die in het kader van het testen een bepaalde mate van uniformiteit hebben, maar binnen de functiepuntanalyse als uniek worden aangemerkt. Binnen de functiepuntanalyse betekent uniek zijn:

- Een unieke combinatie van gegevensverzamelingen ten opzichte van de andere invoerfuncties.
- Geen unieke combinatie van gegevensverzamelingen maar wel een andere logische manier van verwerken (bijvoorbeeld het op een andere manier bijwerken van een gegevensverzameling).

Daarnaast is het zo dat binnen een informatiesysteem functies bestaan die in het kader van functiepuntanalyse als volledig uniform worden aangemerkt en daarom geen functiepunten krijgen toebedeeld, maar binnen het testen wel moeten worden meegeteld aangezien deze functies wel moeten worden getest. Het gaat hierbij om klonen en dummy's.

Berekeningswijze

De factor (A_f) wordt bepaald door de som van de waarden van de eerste vier functieafhankelijke variabelen (gebruikersbelang, gebruiksintensiteit, systeemdoorwerking en complexiteit) te bepalen en deze te delen door 20 (de nominale waarde). Het resultaat van deze bewerking dient vervolgens te worden vermenigvuldigd met de waarde van de factor uniformiteit. De factor A_f wordt per functie bepaald.

$$\mathbf{A_f = ((Gb + Gi + Sd + C) / 20) * U}$$

A_f = wegingsfactor van de functie-afhankelijke factoren

Gb = gebruikersbelang

Gi = gebruiksintensiteit
Sd = systeemdoorwerking
C = complexiteit
U = uniformiteit

Standaard functies

Indien binnen de functiepuntentelling de functies foutmeldingen, helpschermen en/of menustructuur voorkomen, hetgeen veelal het geval is, dienen deze als volgt te worden gewaardeerd:

Functie	FP'en	Gb	Gi	Sd	C	U	A_f
Foutmeldingen	4	6	8	4	3	1	1,05
Helpschermen	4	6	8	4	3	1	1,05
Menustructuur	4	6	8	4	3	1	1,05

Rekenvoorbeeld (2): Bepaling van de functie-afhankelijke variabelen (Af)

	Registreren	Verwerken
Gebruikersbelang	6	12
Gebruiksintensiteit	8	2
Systeemdoorwerking	2	2
Complexiteit	3	6
Uniformiteit	1	1
$\mathbf{A_f}$ =	19/20 * 1 = 0,95	22/20 * 1 = 1,10

(In dit voorbeeld wordt er van uitgegaan dat de waardering van de factoren systeemdoorwerking en complexiteit voor de FPA-functies binnen een gebruikersfunctie identiek zijn.)

Dynamische meetbare kwaliteitsattributen

Hierna wordt aangegeven hoe de eisen die worden gesteld ten aanzien van de dynamisch meetbare kwaliteitsattributen worden meegenomen binnen de testpuntanalyse. Met betrekking tot de dynamisch meetbare kwaliteitsattributen wordt in TPA onderscheid gemaakt tussen kwaliteitsattributen die expliciet en/of impliciet kunnen worden gemeten. Dynamisch expliciet kunnen worden gemeten:

- functionaliteit
- beveiliging
- bruikbaarheid / inpasbaarheid
- performance
- portabiliteit.

Voor ieder kwaliteitsattribuut dient de zwaarte van de kwaliteitseisen, in het kader van de uit te voeren test door middel van een cijfer, eventueel per deelsysteem, te worden gewaardeerd.

Weging

0 niet van belang, wordt daarom niet gemeten
3 lage kwaliteitseisen: er dient in de test wel aandacht aan te worden gegeven
4 normale kwaliteitseisen (veelal van toepassing indien het informatiesysteem betrekking heeft op een ondersteunend proces)
5 hoge kwaliteitseisen (veelal van toepassing indien het informatiesysteem betrekking heeft op een primair proces)
6 extreem hoge kwaliteitseisen.

De kwaliteitsattributen die dynamisch expliciet worden gemeten hebben de volgende wegingsfactoren:

Functionaliteit	wegingsfactor 0,75
Beveiliging	wegingsfactor 0,05
Bruikbaarheid	wegingsfactor 0,10
Performance	wegingsfactor 0,05
Portabiliteit	wegingsfactor 0,05

Er dient te worden vastgesteld welke relevante (in de teststrategie onderkende) kwaliteitsattributen dynamisch impliciet zullen worden getest. Een uitspraak over deze kwaliteitsattributen kan worden gedaan door tijdens de testuitvoering statistieken te verzamelen. Bijvoorbeeld performance kan dus expliciet, met behulp van een real-life test, of impliciet, door het verzamelen van statistieken, worden gemeten.

De dynamisch impliciet te meten kwaliteitsattributen dienen te worden gespecificeerd. Vervolgens kan het aantal kwaliteitsattributen worden bepaald.

Per attribuut geldt een weging van 0,02 ten behoeve van Q_d. In principe kan elk kwaliteitsattribuut dynamisch impliciet worden getest.

Berekeningswijze (Q_d)

Per dynamisch expliciet meetbaar kwaliteitsattribuut wordt de waardering die hieraan is gegeven gedeeld door vier (de nominale waarde) en vervolgens vermenigvuldigd met de wegingsfactor. De resultaten die op deze wijze ontstaan bij de dynamisch expliciet meetbare kwaliteitsattributen, worden opgeteld.

Indien er voor gekozen is bepaalde kwaliteitsattributen dynamisch impliciet mee te nemen bij de test, dient de desbetreffende weging (0,02 per attribuut) te worden opgeteld bij het eerder verkregen resultaat (van de dynamisch expliciet meetbare kwaliteitsattributen). Het op deze wijze verkregen getal is de factor Q_d. De factor Q_d wordt in principe eenmalig voor het totale systeem bepaald. Indien de strategie per deelsysteem afwijkend is, dient de factor Q_d echter te worden bepaald per deelsysteem.

Rekenvoorbeeld (3): Bepaling van de dynamisch meetbare kwaliteitsattributen (Qd)

Functionaliteit	5	(5/4) * 0,75 = 0,94
Beveiliging	4	(4/4) * 0,05 = 0,05
Bruikbaarheid	0	(0/4) * 0,10 = 0
Performance	0	(0/4) * 0,05 = 0
Portabiliteit	0	(0/4) * 0,05 = 0

Dynamisch impliciet worden gemeten:

Performance	= 0,02
Zuinigheid	= 0,02
Onderhoudbaarheid	= 0,02

$\mathbf{Q_d} = 0,94 + 0,05 + (3 * 0,02) = 1,05$

Formule dynamische testpunten

Het aantal dynamische testpunten is een optelling van het aantal testpunten per functie. Het aantal testpunten per functie kan worden vastgesteld door hetgeen nu bekend is in de onderstaande formule in te vullen:

$$\mathbf{TP_f = FP_f * A_f * Q_d}$$

TP_f = het aantal testpunten per functie

FP_f = aantal functiepunten per functie

A_f = wegingsfactor van de functie-afhankelijke factoren

Q_d = wegingsfactor van de dynamische kwaliteitsattributen

Rekenvoorbeeld (4): Berekening dynamische testpunten (TPf)

	FP_f	*	A_f	*	Q_d	=	TP_f
Registreren	18		0,95		1,05	=	18
Verwerken	12		1,10		1,05	=	14 +
Totaal aantal dynamische testpunten							32

11.8.3 Statische testpunten

Het aantal statische testpunten is met name afhankelijk van de statisch te testen kwaliteitsattributen (de factor Q_s). Daarnaast wordt het aantal statische testpunten beïnvloed door het totaal aantal functiepunten van het systeem. Een statische beoordeling van een omvangrijk informatiesysteem kost nu eenmaal meer tijd dan een beoordeling van een eenvoudig informatiesysteem.

Van de relevante kwaliteitsattributen dient te worden vastgesteld of zij al dan niet statisch zullen worden getest. Een uitspraak over deze kwaliteitsattributen wordt gegeven door het toepassen van een checklist. In principe kunnen alle kwaliteitsattributen met behulp van een checklist statisch worden getest. Bijvoorbeeld beveiliging kan dus dynamisch, met behulp van een semantische test, en/of statisch, door het beoordelen van de beveiligingsmaatregelen op basis van een checklist, worden gemeten.

Berekeningswijze (Q_s)

Indien ervoor wordt gekozen een bepaald kwaliteitsattribuut mee te nemen bij de test krijgt de factor Q_s de waarde 16. Voor elk volgende statisch te meten kwaliteitsattribuut die wordt meegenomen wordt de waarde van Q_s met 16 verhoogd.

Rekenvoorbeeld (5): Berekening statische testpunten (Qs)

Statisch (m.b.v. een checklist) worden gemeten:
Continuïteit = 16
Q_s = 16

11.8.4 Totaal aantal testpunten

Het aantal testpunten van het totale systeem kan worden vastgesteld door hetgeen nu bekend is in de onderstaande formule in te vullen:

$$\mathbf{TP = \sum TP_f + ((FP * Q_s) / 500)}$$

TP = het aantal testpunten van het totale systeem
$\sum TP_f$ = de som van het aantal testpunten per functie (dynamische testpunten)
FP = aantal functiepunten van het totale systeem (minimale waarde 500)
Q_s = wegingsfactor van de statische kwaliteitsattributen

Rekenvoorbeeld (6): Berekening totaal aantal testpunten (TP)

TP = 32 + ((500 * 16) / 500) = 48

11.8.5 Primaire testuren

Uit de in de vorige paragraaf genoemde formule ontstaat het totaal aantal testpunten. Het totaal aantal testpunten is een maatstaf voor de omvang van de primaire testactiviteiten. Deze primaire testpunten worden vermenigvuldigd met de vaardigheidsfactor en de omgevingsfactor om tot de primaire testuren te komen. Het aantal primaire testuren is de tijd die nodig is om de testactiviteiten van de fasen Voorbereiding, Specificatie, Uitvoering en Afronding van het TMap-faseringmodel uit te voeren.

Vaardigheidsfactor

De vaardigheidsfactor geeft aan hoeveel uur testen per testpunt nodig is. Een hogere vaardigheidsfactor vereist dus een groter aantal uren testen.

De productiviteit waarmee het testobject op basis van de teststrategie wordt getest, is vooral afhankelijk van de kennis en kunde van de uitvoerenden. Ook is de volledigheid van inzet (fulltime/parttime) belangrijk. Testende gebruikers die slechts een deel van de werkdag voor testwerk ingezet worden, hebben veel omschakelmomenten tussen dagelijks werk en testwerk en daardoor veelal een verminderde productiviteit.

In de praktijk worden de volgende kengetallen per testpunt gebruikt:

- 1-2 uur voor een tester, afhankelijk van kennis en kunde
- 2-4 uur voor een gebruiker, afhankelijk van ervaring.

De vaardigheidsfactor kan uiteraard per organisatie, en zelfs daarbinnen per afdeling/persoon, verschillen. Een vaardigheidsfactor kan worden verkregen door een analyse van reeds gerealiseerde testprojecten. Om een dergelijke analyse te kunnen maken is het noodzakelijk om te beschikken over ervaringscijfers van de gerealiseerde testprojecten.

Rekenvoorbeeld (7): Vaardigheidsfactor

Voor de betreffende organisatie geldt een vaardigheidsfactor van 1,2.
V = 1,2

Omgevingsfactor

Het aantal benodigde testuren per testpunt wordt niet alleen beïnvloed door de vaardigheidsfactor, maar ook door de omgevingsfactor.

Voor het berekenen van de omgevingsfactor is een aantal omgevingsvariabelen onderkend. Hieronder worden omgevingsvariabelen beschreven inclusief de bijbehorende waarderingen. Het is ook hier slechts mogelijk voor een van de beschreven waarden te kiezen. Indien er te weinig informatie beschikbaar is om een bepaalde variabele te classificeren dan dient deze de nominale waarde (vet gedrukt) te krijgen.

Testtools

De factor testtools betreft de mate waarin de primaire testactiviteiten door geautomatiseerde testtools worden ondersteund. Testtools kunnen ertoe bijdragen dat een deel van de testactiviteiten automatisch en daardoor sneller wordt uitgevoerd. De beschikbaarheid van testtools is echter geen garantie hiervoor. Het gaat om het doelmatig gebruik ervan.

Weging

1 er wordt bij de test gebruik gemaakt van ondersteunende tools t.b.v. testspecificatie én er wordt een tool gebruikt ten behoeve van ‘record & playback’
2 er wordt bij de testuitvoering gebruik gemaakt van ondersteunende tools t.b.v. testspecificatie of er wordt van een tool met ‘record & playback’ mogelijkheden gebruik gemaakt
4 er zijn geen testtools aanwezig.

Voorgaande test

Bij deze factor is de kwaliteit van de eerder uitgevoerde test van belang. Bij het begroten van een acceptatietest betreft dit een systeemtest en bij het begroten van een systeemtest de ontwikkeltest. De kwaliteit van de voorgaande test is mede bepalend voor de hoeveelheid functionaliteit die eventueel in beperktere mate getest kan worden en bovendien voor de doorloop van de testuitvoering. Bij een hogere kwaliteit van de voorgaande test treden namelijk minder voortgang belemmerende storingen op.

Weging

2 er is in het kader van de voorgaande test een testplan aanwezig en tevens heeft het testteam inzicht in de concrete testgevallen en testresultaten (test coverage)
4 er is in het kader van de voorgaande test een testplan aanwezig
8 er is geen testplan in het kader van de voorgaande test aanwezig.

Testbasis

Testbasis is gedefinieerd als de kwaliteit van de (systeem)documentatie waarop de uit te voeren test dient te zijn gebaseerd. De kwaliteit van de testbasis is met name van invloed op de benodigde tijd voor de fasen Voorbereiding en Specificatie.

Weging

3 er wordt bij het opstellen van de systeemdocumentatie gebruik gemaakt van standaards en sjablonen, tevens vinden inspecties plaats op de documentatie
6 er wordt bij het opstellen van de documentatie gebruik gemaakt van standaards en sjablonen
12 er wordt bij de systeemontwikkeling geen gebruik gemaakt van standaards en sjablonen

Ontwikkelomgeving

De omgeving waarin de realisatie van het informatiesysteem plaatsvindt. Met name is het hierbij van belang in hoeverre de ontwikkelomgeving reeds het maken van fouten voorkomt c.q. bepaalde dingen afdwingt. Indien bepaalde fouten niet meer kunnen worden gemaakt behoeven deze uiteraard niet meer te worden getest.

Weging

2 de ontwikkelomgeving bezit een groot aantal faciliteiten om het maken van fouten te voorkomen door bijvoorbeeld het uitvoeren van semantische en syntactische controles en het overnemen van parameters
4 de ontwikkelomgeving bezit een beperkt aantal faciliteiten om het maken van fouten te voorkomen door bijvoorbeeld het uitvoeren van syntactische controles en het overnemen van parameters
8 de ontwikkelomgeving bezit geen faciliteiten om het maken van fouten te voorkomen.

Testomgeving

De mate waarin de fysieke testomgeving, waarin de test wordt uitgevoerd, beproefd is. Indien gebruik gemaakt wordt van een veelvuldig beproefde testomgeving zullen minder (ver)storingen plaatsvinden tijdens de fase Uitvoering.

Weging

1 de omgeving is reeds meerdere malen gebruikt voor het uitvoeren van een test
2 er is voor de betreffende test een nieuwe omgeving ingericht, binnen de organisatie is reeds in ruime mate ervaring met soortgelijke omgevingen
4 er is voor de betreffende test een nieuwe omgeving ingericht die voor de organisatie kan worden gekenmerkt als experimenteel.

Testware

De mate waarin tijdens de uit te voeren test gebruik kan worden gemaakt van reeds aanwezige testware. De beschikbaarheid van bruikbare testware is met name van invloed op de hoeveelheid tijd benodigd voor de fase Specificatie.

Weging

1 er is reeds een bruikbare algemene centrale uitgangssituatie (tabellen e.d.) aanwezig alsmede gespecificeerde testgevallen ten behoeve van de uit te voeren test
2 er is reeds een bruikbare algemene centrale uitgangssituatie (tabellen e.d.) aanwezig
4 er is geen bruikbare testware beschikbaar.

Berekeningswijze

De omgevingsfactor (O) wordt bepaald door de som van de waarden van de omgevingsvariabelen (testtools, voorgaande test, testbasis, ontwikkelomgeving, testomgeving en testware) te bepalen en deze te delen door 21 (de nominale waarde). De omgevingsfactor O kan eenmalig voor het totale systeem worden bepaald, maar eventueel ook per deelsysteem.

Rekenvoorbeeld (8): Omgevingsfactor (O)

De diverse omgevingsvariabelen hebben de onderstaande waardering gekregen:

Testtools	4 (geen testtools)
Voorgaande test	4 (er is een testplan van de voorgaande test aanwezig)
Testbasis	3 (documentatiesjablonen en inspecties)
Ontwikkelomgeving	4 (Oracle in combinatie met COBOL)
Testomgeving	1 (beproefde omgeving)
Testware	4 (geen bruikbare testware aanwezig)

O = 20/21 = 0,95

Formule primaire testuren

Het aantal primaire testuren wordt verkregen door het aantal testpunten te vermenigvuldigen met de vaardigheidsfactor en de omgevingsfactor:

PT = TP * V * O

PT = het totaal aantal primaire testuren
TP = het aantal testpunten van het totale systeem
V = vaardigheidsfactor
O = omgevingsfactor

Rekenvoorbeeld (9): Berekening primaire testuren (PT)

PT = 48 * 1,2 * 0,95 = 54,72 (55 uur)

11.8.6 Totaal aantal testuren

Aangezien in elk testproces secundaire activiteiten worden uitgevoerd uit de fase Beheer en uit de fase Inrichting en beheer infrastructuur dient er ten behoeve hiervan nog een opslag plaats te vinden op de primaire testuren. Dit levert dan uiteindelijk het totaal aantal testuren op. Het aantal opslaguren wordt berekend als een percentage van de primaire testuren.

De hoogte van dit opslagpercentage wordt vaak op basis van ervaring door een testmanager of door historische gegevens bepaald. Er zijn ook organisaties die hiervoor een vast percentage hanteren. Vrijwel altijd bevindt dit percentage zich in het gebied van 5% tot 20%.

In het geval ervaring, historische gegevens en/of vaste percentages ontbreken, kan op volgende wijze een opslagpercentage worden geschat. Hierbij wordt als uitgangspunt een standaard (nominale) opslagpercentage van 12% gehanteerd. Vervolgens moet er worden gekeken naar factoren waardoor dit percentage hoger of lager kan worden. Voorbeelden van factoren zijn:

- Teamgrootte
- Beheerinstrumenten
- Permanente testorganisatie

Deze factoren worden hieronder toegelicht. Aangezien er een grote diversiteit aan testprojecten bestaat, is bij de bepaling van de invloed van deze factoren op het percentage niet gekozen voor schijnzekere absolute percentagecijfers, maar voor het aangeven of de invloed percentageverhogend of -verlagend is.

Teamgrootte

De teamgrootte betreft het aantal leden van het testteam (inclusief testmanager en eventuele testbeheerder). Een groot team leidt meestal tot meer ‘overhead’ en daarmee tot een hoger opslagpercentage. Een klein testteam echter leidt tot verlaging van het percentage:

- Verlaging
  Testteam bestaat uit maximaal 4 personen.
- Neutraal
  Testteam bestaat uit 5 - 9 personen.
- Verhoging
  Testteam bestaat uit minimaal 10 personen.

Beheerinstrumenten

Bij beheerinstrumenten wordt bekeken in hoeverre van geautomatiseerde hulpmiddelen gebruik wordt gemaakt tijdens het testproces met betrekking tot de Beheer en Inrichting en beheer infrastructuur activiteiten. Voorbeelden van deze hulpmiddelen zijn een geautomatiseerd:

- planningssysteem
- voortgangsbewakingssysteem
- bevindingenbeheersysteem
- testwarebeheersysteem.

Als er weinig gebruik wordt gemaakt van geautomatiseerde hulpmiddelen zullen bepaalde activiteiten handmatig moeten worden uitgevoerd. Dit leidt tot een verhoging van het opslagpercentage. Als er veel gebruik wordt gemaakt van geautomatiseerde hulpmiddelen leidt dat tot een verlaging van het percentage:

- Verlaging
  Er wordt van minimaal 3 geautomatiseerde hulpmiddelen gebruik gemaakt.
- Neutraal
  Er wordt van 1 - 2 geautomatiseerde hulpmiddelen gebruik gemaakt.
- Verhoging
  Er wordt geen gebruik gemaakt van geautomatiseerde hulpmiddelen.

Permanente testorganisatie

Er zijn vele vormen van permanente testorganisaties (paragraaf 8.3). Als een organisatie over één van de vormen van een permanente testorganisatie beschikt, wordt in een testtraject die daarvan gebruik maakt, vaak doorlooptijdverkorting, kostenbesparing en/of verbetering van de kwaliteit gerealiseerd:

- Verlaging

Testteam maakt gebruik van diensten van een permanente testorganisatie.
- Neutraal
  Testteam maakt geen gebruik van diensten van een permanente testorganisatie.

Rekenvoorbeeld (10): Bepaling opslag Beheer en Inrichting en beheer infrastructuur (B en I&B I)

Uit historische gegevens blijk het opslagpercentage voor vergelijkbare testprojecten rond de 15% te schommelen. De testmanager besluit dit percentage over te nemen.

Opslagpercentage B en I&B I = 15%

Berekeningswijze

Met het opslagpercentage wordt op basis van het aantal primaire testuren de opslag (in uren) berekend.

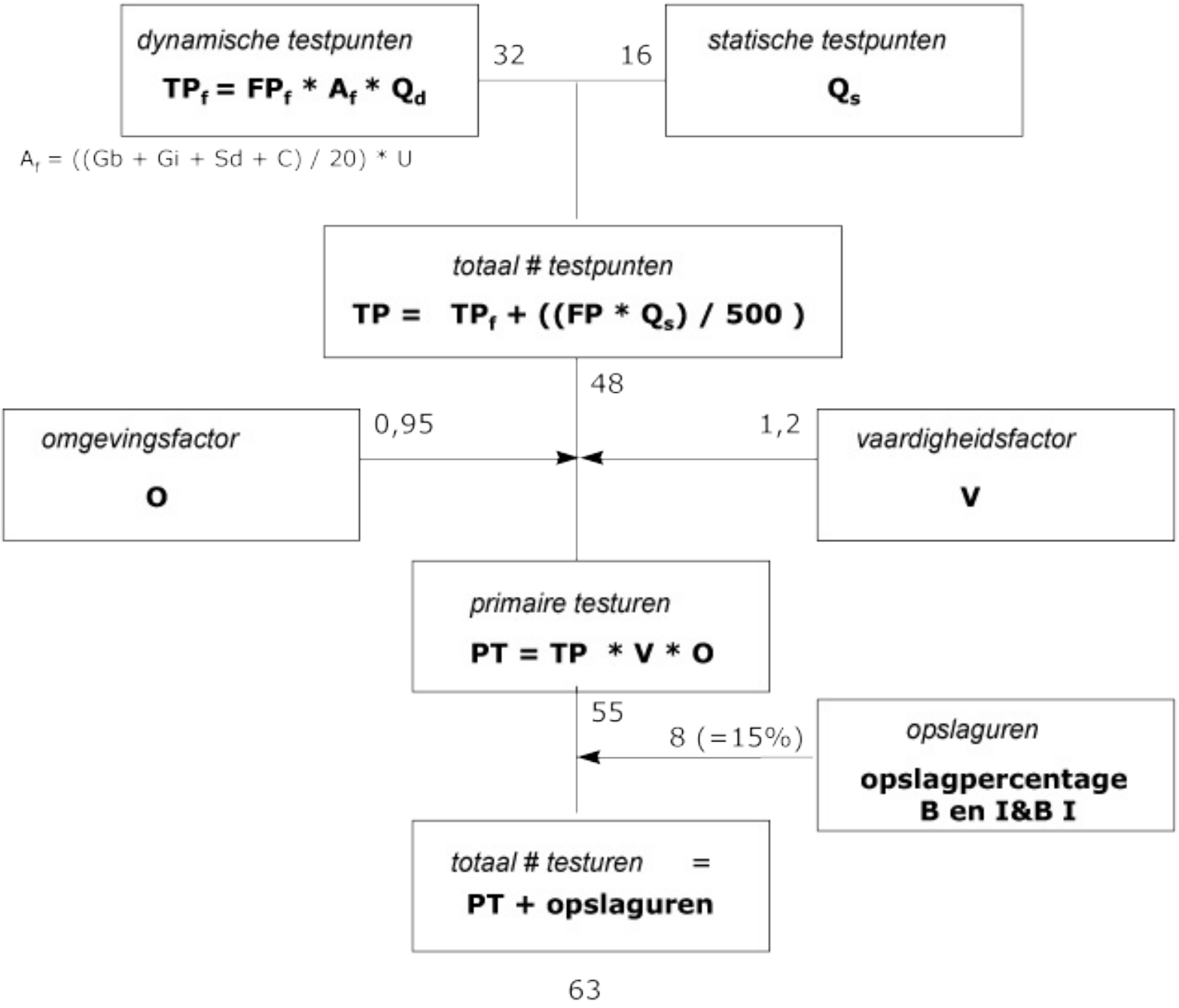
Het totaal aantal testuren wordt tenslotte verkregen door de berekende opslag voor Beheer en Inrichting en beheer infrastructuur toe te voegen aan het totaal aantal primaire testuren.

Rekenvoorbeeld (11): Berekening totaal aantal testuren

Primaire testuren 55
Opslag B en I&B I 55 * 0,15 = 8,25 (afgerond: 8)

Totaal aantal uren 55 + 8 = 63

In figuur 11.4 is het TPA rekenvoorbeeld in zijn geheel weergegeven.



Figuur 11.4 : Schematische weergave rekenvoorbeeld.

11.8.7 Verdeling over de fasen

Het testproces is bij gebruik van TMap onderverdeeld in een zevental fasen en menig opdrachtgever zal naast de begroting voor het gehele testproces tevens geïnteresseerd zijn in de begroting per fase.

TPA geeft een begroting voor het gehele testproces exclusief het opstellen van het testplan (fase Planning).

Voor de fasen Beheer en Inrichting en beheer infrastructuur wordt in principe het aantal uren begroot dat, met behulp van het opslagpercentage, op basis van het aantal primaire testuren was berekend (opslaguren). Deze opslaguren moeten over de twee fasen worden verdeeld.

De primaire testuren worden verdeeld over de overige fasen (Voorbereiding, Specificatie, Uitvoering en Afronding). De verdeling van de primaire testuren over de fasen kan uiteraard per organisatie, en zelfs daarbinnen, verschillen. Een voor de organisatie van toepassing zijnde verdeling kan worden verkregen door een analyse van reeds gerealiseerde testprojecten. Om een dergelijke analyse te kunnen maken is het noodzakelijk om te beschikken over ervaringscijfers van reeds gerealiseerde testprojecten.

Verdeling van de primaire testuren

Praktijkervaringen met testpuntanalyse in combinatie met TMap leveren de onderstaande verdeling van de testinspanning over de diverse fasen op:

Voorbereiding	10%
Specificatie	40%
Uitvoering	45%
Afronding	5%

Voor andere verdelingen op basis van praktijkervaringen zie paragraaf 11.2.

11.8.8 TPA in een vroegtijdig stadium

Veelal moet reeds in een vroegtijdig stadium een projectbegroting voor het testen worden gemaakt. Het is dan niet mogelijk om factoren zoals complexiteit, doorwerking en dergelijke te bepalen als gevolg van het ontbreken van gedetailleerde functionele specificaties. Er zijn echter aanpakken die vaak wél kunnen worden toegepast voor het uitvoeren van een globale testpuntanalyse. Door gebruik te maken van één van onderstaande aanpakken kan het totaal aantal (bruto) functiepunten worden geschat:

- voer op basis van zeer globale specificaties een zogenaamde globale functiepuntanalyse uit
- bepaal het aantal functiepunten door het aantal TOM te bepalen (zie voor nadere toelichting paragraaf 11.3).

Vervolgens wordt in het kader van een globale testpuntanalyse één functie gedefinieerd. Deze functie heeft de omvang heeft van het totaal aantal vastgestelde (bruto) functiepunten. Alle functieafhankelijke factoren (gebruikersbelang, gebruiksintensiteit, complexiteit, doorwerking en uniformiteit) krijgen in principe de neutrale waarde, waardoor A_f de waarde één krijgt. Vervolgens kan een testpuntanalyse worden uitgevoerd zoals beschreven in de vorige paragrafen. Bij het vaststellen van de omgevingsfactor zullen veelal aannames moeten worden gedaan. Het is van belang bij het presenteren van de testbegroting ook duidelijk deze aannames te vermelden.

Noot

1 COSMIC: COmmon Software Measurement International Consortium

12 Bevindingenbeheer

12.1 Inleiding

Veel partijen zien het vinden van fouten als het doel van testen. Hoewel duidelijk mag zijn dat het doel van testen meer is dan dat, namelijk het geven van informatie en advies over risico's en kwaliteit, blijft het feit dat het vinden van fouten één van de belangrijkste activiteiten van testen is.

Binnen testen gebruikt men echter de term bevindingen in plaats van fouten, omdat dit veel neutraler is dan 'fout', dat een negatieve bijklank heeft.

Definitie

Een bevinding is een geconstateerd verschil tussen de verwachting of voorspelling en de feitelijke uitkomst.

Een bevinding kan gedaan worden op verschillende objecten, zoals de testbasis (bij de detailintake), het te testen systeem, de test infrastructuur, enzovoort.

Testers moeten beseffen dat ze a) andermans werk aan het beoordelen zijn en b) dat het uiteindelijke product een resultaat van samenwerking is tussen alle partijen. Het is vriendelijker naar de andere partijen toe om een afwijking te constateren tussen wat de software doet en wat de tester verwacht op basis van de beschikbare informatie, dan dat de tester gelijk roept dat hij een fout gevonden heeft. Dit laatste werkt polariserend en wordt snel een discussie over wie de fout heeft gemaakt in plaats van een discussie over hoe de fout het beste opgelost kan worden. Een andere reden om voorzichtig met het woord 'fout' te zijn, is dat de oorzaak van de bevinding regelmatig niet bij de ontwikkelaar blijkt te liggen, maar bij de tester zelf. In een situatie waarbij ontwikkelaars en testers tegenover elkaar staan in plaats van naast elkaar, kan een aantal onterecht gerapporteerde fouten de geloofwaardigheid van de testers geheel onderuit halen.

Het administreren en bewaken van de bevindingen betreft ook het oplossen ervan en is daarmee feitelijk een projectaangelegenheid is en niet specifiek een zaak van de testers. Wel zijn testers hier het meest bij betrokken. Een goede administratie moet de levensloop van een bevinding kunnen bewaken en daarnaast diverse overzichten kunnen leveren. Deze overzichten worden onder andere gebruikt om gefundeerde kwaliteitsuitspraken te doen. Het beheer van de bevindingen wordt soms belegd bij een aparte rol: bevindingenbeheerder (zie paragraaf 16.3 "Rollen niet als functie beschreven").

Vanuit het testproces zijn de testers de indieners van bevindingen en controleren zij de oplossing ervan. De testmanager communiceert met de overige partijen over (de afhandeling van) de bevindingen. Ook wordt wel gekozen om deze taak te beleggen bij een aparte rol in het team: de intermediair. Deze heeft als doel het adequaat kanaliseren van de bevindingen en de bijbehorende oplossingen. De intermediair onderhoudt hierover op uitvoerend niveau de externe contacten. De intermediair heeft overzicht over alle bevindingen en fungeert als doorgeefluik en controlepost van enerzijds de bevindingen en anderzijds de oplossingen. Voordelen hiervan zijn dat de kwaliteit van de bevindingen en oplossingen beter wordt bewaakt en dat de communicatie wordt gestroomlijnd.

Het heeft grote voordelen één enkele bevindingenadministratie en -procedure voor het gehele project of de gehele lijnafdeling in te richten. Alle betrokken partijen, dus ontwikkelaars, gebruikers, testers, QA'ers, enzovoort, kunnen zowel hun bevindingen als oplossingen hierin kwijt. De communicatie over de afhandeling van de bevindingen wordt hiermee sterk vereenvoudigd. Ook biedt een centrale administratie extra mogelijkheden om informatie te verkrijgen. De autorisaties vormen hierbij een aandachtspunt: het moet niet mogelijk zijn dat bevindingen door hiertoe niet bevoegde personen kunnen worden aangepast of afgesloten.

Dit verdere hoofdstuk is ingedeeld in een aantal paragrafen die gezamenlijk de levensloop van een bevinding weergeven:

- het doen van een bevinding (paragraaf 12.2);
- het rapporteren van een bevinding (paragraaf 12.3);
- de procedure rond het afhandelen van de bevinding (paragraaf 12.4).

12.2 Een bevinding doen

Gedurende vrijwel het gehele testproces kunnen bevindingen worden gedaan. De nadruk ligt echter op de fasen Voorbereiding, Specificatie en Uitvoering. Omdat in de fasen Voorbereiding en Specificatie normaal gesproken het testobject nog niet gebruikt wordt, doen de testers in deze fasen bevindingen op de testbasis. Tijdens de fase Uitvoering constateren de testers verschillen tussen de feitelijke en verwachte werking van het testobject. De oorzaak van deze bevindingen kan dan echter ook nog steeds liggen in de testbasis.

Onderstaand zijn de stappen beschreven die de tester moet doorlopen bij het constateren van een bevinding:

- Verzamel bewijsmateriaal;
- Reproduceer de bevinding;
- Controleer op eigen fouten;
- Bepaal vermoedelijke externe oorzaak;
- Isoleer de oorzaak (optioneel);
- Generaliseer de bevinding;
- Vergelijk met andere bevindingen;
- Schrijf bevindingrapport;
- Laat reviewen.

De stappen staan in globale volgorde van uitvoering, maar het is zeer goed mogelijk bepaalde stappen in een andere volgorde of parallel uit te voeren. Als de tester bijvoorbeeld gelijk ziet dat de bevinding al eerder in dezelfde test gedaan is, hoeven alle tussenliggende stappen niet meer doorlopen te worden.

Verzamel bewijsmateriaal

Op een bepaald moment geeft het testobject een andere reactie dan de tester verwacht of constateert de tester dat de testbasis een onduidelijkheid, inconsistentie of omissie bevat: een bevinding. De eerste stap is om het bewijs van deze afwijking vast te leggen. Dit kan bijvoorbeeld bij testuitvoering door een screendump of memorydump te maken, uitvoer te printen, een kopie van de database-inhoud te maken of aantekeningen te maken.

De tester moet ook kijken naar andere plaatsen waar het resultaat van de afwijking zichtbaar zou kunnen zijn. Dit kan door bijvoorbeeld bij onverwacht resultaat in een Toevoeg-functie te kijken hoe de gegevens in de database zijn opgeslagen en hoe de gegevens in de Raadpleeg-functie getoond worden.

Als de bevinding op een onderdeel van de testbasis betrekking heeft, moet gekeken worden naar andere onderdelen van de testbasis die een relatie hebben met het betreffende onderdeel.

Reproduceer de bevinding

Bij bevindingen tijdens testuitvoering is de volgende stap dat gekeken wordt of de bevinding reproduceerbaar is door het testgeval nog een keer uit te voeren. De tester is nu veel alerter op afwijkend systeemgedrag. Bovendien helpt het nog een keer uitvoeren om eventuele testuitvoeringsfouten te herkennen. Is de bevinding reproduceerbaar, dan gaat de tester verder met de volgende stappen. Is de bevinding niet reproduceerbaar en is het vermoedelijk geen testuitvoeringsfout, dan wordt het moeilijker. De tester voert het testgeval nog een keer uit. De tester geeft dan in het bevindingrapport duidelijk aan dat de bevinding niet reproduceerbaar is of dat de bevinding in 2 van de 3 gevallen optreedt. De kans is reëel dat de ontwikkelaars aan een niet-reproduceerbare bevinding weinig of geen tijd zullen besteden. Het nut van het toch als bevinding indienen, is dat dit een geschiedenis opbouwt van niet-reproduceerbare bevindingen. Blijkt een niet-reproduceerbare bevinding vaak voor te komen, dan kan de verwachting worden uitgesproken dat deze in productie ook regelmatig zal optreden en daarom opgelost moet worden.

Voorbeeld

Tijdens een systeemtest crashte het systeem op niet-reproduceerbare wijze een paar keer per dag. Het testteam meldde dit elke keer in een bevinding, maar het ontwikkelteam stond onder tijdsdruk, besteedde geen aandacht aan deze bevindingen en deed het af als instabiliteit van het gebruikte ontwikkelpakket. Door het overleggen van het grote aantal niet-reproduceerbare bevindingen en het aangeven dat een negatief vrijgaveadvies het gevolg zou zijn, zijn ze uiteindelijk overtuigd. Binnen relatief korte tijd vonden ze de oorzaak (een programmeerfout) en losten het probleem op.

Uitgediept

In sommige gevallen, zoals bij performancetests en testen van batchprogrammatuur, kost het onevenredig veel tijd om de test opnieuw uit te voeren. In die gevallen wordt de test om te kijken of de bevinding reproduceerbaar is, niet herhaald.

Controleer op eigen fouten

De tester gaat kijken naar de mogelijke oorzaak van de bevinding. Hiervoor kijkt hij als eerste naar een mogelijke interne oorzaak. De bevinding kan bijvoorbeeld veroorzaakt zijn door een fout in:

- de testspecificatie of (centrale) uitgangssituatie;
- de testomgeving of testtools;
- de testuitvoering;
- de beoordeling.

Hierbij moet de tester er ook rekening mee houden dat de testresultaten verstoord kunnen zijn door de resultaten van een andere test van een collegatester.

Als de oorzaak intern ligt, moet de tester deze (laten) oplossen, bijvoorbeeld door de testspecificatie aan te passen. Vervolgens herhaalt de tester het testgeval, hetzij nog in dezelfde testronde, hetzij in de volgende testronde.

Uitgediept

Testomgevingen en testtools staan normaal gesproken onder beheer van de testers.

Bevindingen hierop die binnen het team opgelost kunnen worden, horen tot de interne bevindingen. Wanneer de oplossing van de bevinding van buiten het team komt, is het een externe bevinding.

Bepaal vermoedelijke externe oorzaak

Als de oorzaak niet bij het testen zelf ligt, moet extern gezocht worden. Externe oorzaken kunnen bijvoorbeeld zijn:

- testbasis;
- testobject (software, maar ook documentatie zoals gebruikershandleiding of AO-procedures);
- testomgeving en testtools.

De tester moet de oorzaak zo goed mogelijk achterhalen, omdat dit helpt bij het bepalen wie de bevinding moet oplossen en later bij het ontdekken van kwaliteitstrends.

Omdat de tester zijn testgeval vergelijkt met het testobject is er de neiging om bij een afwijking het testobject als eerste oorzaak aan te wijzen. Toch moet de tester verder zoeken: ligt de oorzaak niet in de testbasis? Is er misschien een onderlinge inconsistentie tussen de verschillende vormen van testbasis?

Regelmatig gebruikt de tester behalve de formele testbasis (zoals bijvoorbeeld het functioneel ontwerp of de requirements) andere, minder tastbare vormen van testbasis. Dergelijke vormen kunnen zijn de onderlinge consistentie van de schermen en gebruikersinterface, de vergelijking met vorige releases of concurrerende producten en de verwachtingen van de gebruikers. Zie hiervoor ook paragraaf 6.5 “Fase Voorbereiding”. Bij het beschrijven van de bevinding is dan belangrijk om aan te geven welke afwijkende vorm van testbasis gebruikt is én of het testobject wel of niet overeenstemt met de formele en beschreven testbasis, zoals de requirements of het functionele ontwerp. In het geval dat testobject en formele testbasis overeenstemmen, is de oorzaak van de bevinding een inconsistentie tussen de informele en formele testbasis en niet het testobject.

Voorbeeld

Tijdens een exploratory test ontdekt de tester dat de plaats van de bedieningsknoppen op veel schermen verschilt.

Verder uitzoekwerk toont aan dat de oorzaak bij de schermontwerpen ligt en niet bij het programmeren. De tester dient de bevinding in met als oorzaak de testbasis.

Een externe bevinding wordt altijd formeel beheerd. Dit kan met een in onderstaande paragrafen beschreven bevindingrapport en -procedure. Bij reviews kan gekozen worden voor een lichtere vorm waarbij de bevindingen in een reviewdocument worden gegroepeerd en overhandigd aan de oplosser, zie hiervoor ook hoofdstuk 15 “Toetstechnieken”.

Isoleer de oorzaak (optioneel)

Hoewel de vermoedelijke oorzaak vaak al is aan te wijzen, is dit in het geval van een bevinding op het testobject of de testomgeving vaak nog onvoldoende precies voor de oplosser. De tester kijkt daarom naar omliggende testgevallen die al of niet met goed gevolg doorlopen zijn. Ook brengt hij zo nodig wat variaties aan op het testgeval en voert deze opnieuw uit. Hiermee is vaak een preciezere oorzaak aan te wijzen of kunnen de omstandigheden waaronder de bevinding optreedt specifieker gemaakt worden. Deze stap is optioneel omdat dit op het grensvlak ligt hoe ver de tester ten opzichte van de ontwikkelaar moet gaan om de oorzaak van een bevinding uit te zoeken. Het is belangrijk om hier vooraf afspraken over te maken met de ontwikkelaars. Anders ontstaat er later discussie over het uitvoeren van (extra) analyses op het moment dat het testen op het kritieke pad ligt.

Generaliseer de bevinding

Als de oorzaak voldoende duidelijk lijkt, kijkt de tester of er nog andere plaatsen zijn waar de bevinding kan optreden. Bij testobjectbevindingen voert de tester bijvoorbeeld soortgelijke testgevallen uit op andere plaatsen in het testobject. Dit moet wel in overleg met de andere testers om te voorkomen dat deze tests de tests van collega’s verstoren.

Ook bij testbasisbevindingen kijkt de tester op soortgelijke plaatsen in de testbasis (“In het functioneel ontwerp is voor functie A de controle op overlappende periodesn verkeerd gespecificeerd. Hoe zit het met andere functies die deze controle hebben?”).

Voorbeeld

Tijdens een vrijdagmiddagtest leverde het parallel wijzigen van hetzelfde gegeven door twee gebruikers in functie X een bevinding op. Verder testen op andere functies leerde dat het multi-user mechanisme structureel verkeerd was ingebouwd.

De tester hoeft hierbij geen volledigheid na te streven, maar moet een indruk kunnen geven van de omvang en ernst van de bevinding. Als de bevinding structureel is, is het aan de oplosser om deze structureel op te lossen. Deze stap heeft ook als doel om een zo goed mogelijk beeld op te bouwen van de schade die de bevinding kan veroorzaken in productie.

Vergelijk met andere bevindingen

Voordat de tester het bevindingrapport schrijft, kijkt hij of de bevinding al gedaan is. Dit kan op dezelfde versie van het testobject gedaan zijn door een collega-tester vanuit een andere test. Het kan ook zijn dat de bevinding in een eerdere release al gedaan is.

Hiervoor raadpleegt de tester de bevindingenadministratie, zijn collega-testers, de testmanager, bevindingenbeheerder of intermediair.

Er is een aantal mogelijkheden:

- De bevinding is al gedaan op hetzelfde onderdeel van de huidige release.
  De bevinding hoeft niet ingediend te worden. In het testuitvoeringsverslag kan bij het testgeval worden verwezen naar de al bestaande bevinding.
- Een soortgelijke bevinding is al gedaan op een ander onderdeel van de huidige release.
  De bevinding moet wel ingediend worden en moet een verwijzing bevatten naar de andere bevinding.
- De bevinding is al gedaan op hetzelfde onderdeel van een vorige release.
  Als de oude bevinding opgelost zou moeten zijn voor deze release, moet afhankelijk van de afspraken de oude

bevinding worden heropend of de bevinding opnieuw worden ingediend met verwijzing naar de oude bevinding. Staat de oude bevinding nog open, dan hoeft de tester geen nieuwe bevinding in te dienen.

Tips

- De testmanager, bevindingenbeheerder of intermediair doet er goed aan frequent een overzicht van gedane bevindingen naar de testers te sturen. Zo blijven de testers op de hoogte van gedane bevindingen en worden ze op ideeën gebracht om in hun eigen test naar soortgelijke bevindingen te kijken. Alternatief is dat de testers zelf regelmatig de bevindingenadministratie raadplegen op de gedane bevindingen.
- Het is ook mogelijk om af te spreken dat de testers *niet* letten op dubbele bevindingen. Dit wordt gedaan om de voortgang van testuitvoering niet te verstoren. Het controleren op dubbele bevindingen wordt dan door de intermediair gedaan die daarmee het recht heeft om dubbele bevindingen samen te voegen. Ingeval van twijfel moet de intermediair uiteraard wel overleggen met de betrokken testers.

Schrijf bevindingrapport

De tester legt de bevinding vast in de bevindingenadministratie door middel van een bevindingrapport. Hierin beschrijft hij de bevinding en vult de benodigde velden van het rapport in, zie paragraaf 12.3.

De beschrijving van de bevinding moet helder, eenduidig en to-the-point zijn.

Hierbij moet de toonzetting neutraal zijn. De tester moet zich opstellen als onpartijdig en beseffen dat hij slecht nieuws brengt. Sarcasme, cynisme of overdrijving zijn vanzelfsprekend uit den boze.

Bij voorkeur maakt de tester duidelijk wat de gevolgen zijn bij het niet oplossen van de bevinding, ofwel wat de schade is in productie. Dit bepaalt de kans dat de bevinding ook opgelost wordt. In sommige gevallen is de schade heel duidelijk (“facturen worden verkeerd berekend”) en behoeft weinig toelichting, in andere gevallen is dit onduidelijker (“verkeerd kleurgebruik schermen”) en moet de tester goed aangeven wat de consequenties kunnen zijn (“afwijking van bedrijfsstandaards betekent dat afdeling Externe Communicatie vrijgave van de applicatie kan tegenhouden”). Overigens is het voor de tester lang niet altijd mogelijk de potentiële schade goed in te schatten omdat hem hiertoe de kennis ontbreekt. De uiteindelijke verantwoordelijkheid voor het goed inschatten van de schade ligt dan ook bij (de vertegenwoordigers van) de gebruikers en opdrachtgever in het verderop te behandelen bevindingenoverleg.

Een lastige vraag is altijd hoeveel informatie de beschrijving moet bevatten.

De richtlijn hiervoor is dat de oplosser redelijkerwijs zonder verdere toelichting van de tester de bevinding kan oplossen.

Uitgediept

‘Redelijkerwijs’ in bovenstaande zin is een lastig begrip. Het liefst wil de ontwikkelaar dat de tester aangeeft welk statement in de programmatuur fout is. Dit is echter debuggen en valt onder verantwoordelijkheid van de ontwikkelaar. Er moet dan ook voorkomen worden dat de tester regelmatig bij de programmeur gaat zitten om samen de oorzaak van de bevinding op te sporen. Dit wijst eerder op slecht geschreven bevindingen dan op een coöperatieve testhouding. De tester is op dat moment niet meer met testwerkzaamheden bezig, terwijl de testmanager dit wel van hem verwacht. Als dit regelmatig gebeurt, maakt dit de planning van het testproces onbeheersbaar.

In sommige gevallen doet de tester een heleboel kleine bevindingen op een bepaald onderdeel, bijvoorbeeld een scherm. Er is dan de neiging om de administratie wat simpeler te houden door al deze bevindingen te groeperen in één verzamelbevindingrapport. Ook is er om dezelfde reden óf om het aantal bevindingen wat minder te laten lijken soms druk van ontwikkelaars om dit zo te doen. In de meeste gevallen is dit niet aan te bevelen. De kans is dat van zo’n verzamelbevinding een aantal bevindingen wordt opgelost in de eerstvolgende release, een aantal andere in de daaropvolgende release en een aantal helemaal niet wordt opgelost. Het volgen en bewaken van zo’n verzamelbevinding wordt daarmee een administratieve nachtmerrie.

Laat reviewen

Voordat de bevinding formeel de bevindingenprocedure ingaat, laat de tester het rapport reviewen op volledigheid, juistheid en toonzetting. Dit kan gedaan worden door een collega-tester, de testmanager, bevindingenbeheerder of intermediair. Na verwerking van hun commentaar wordt de bevinding formeel ingediend. Afhandeling hiervan verloopt volgens de in paragraaf 12.4 beschreven procedure.

Voor meer informatie over het doen van een bevinding zie [Black 2004].

12.3 Bevindingrapport

Een bevindingrapport is meer dan alleen een beschrijving van de bevinding.

Ook andere gegevens over de bevinding moeten worden vastgelegd (bijvoorbeeld versie van het testobject, naam van de tester). Om dat op gestructureerde wijze te doen, wordt een bevindingrapport vaak opgedeeld in meerdere ‘velden’. In deze velden kunnen de verschillende gegevens worden vastgelegd. Deze velden zijn nodig om de bevinding te kunnen beheren en om zinvolle informatie uit de administratie te kunnen halen. De belangrijkste redenen om afzonderlijke velden op te nemen in plaats van één groot vrij tekstveld zijn dat:

- de velden afdwingen dat de bevindingeninformatie zo compleet mogelijk wordt ingevoerd;
- het maken van rapportages over selecties van de bevindingen mogelijk is.

Zo kan bijvoorbeeld gemakkelijk worden geselecteerd op alle openstaande bevindingen, op alle bevindingen met als oorzaak de testomgeving of op alle bevindingen op een bepaald onderdeel van het testobject.

Bevindingrapporten kunnen bijna niet meer zonder geautomatiseerde ondersteuning. Dit kan eenvoudig met een spreadsheet of databasepakket, maar er zijn ook diverse freeware of commerciële tools beschikbaar. Deze laatste groep tools heeft vaak als voordeel dat de bevindingenadministratie geïntegreerd is met testwaremanagement en planning- en voortgangsbewaking. Aandachtspunt bij de tools zijn autorisaties. Het moet niet mogelijk zijn dat een ontwikkelaar een bevinding van een tester wijzigt of afsluit, maar het moet wel mogelijk zijn dat de ontwikkelaar een oplossing toevoegt aan de bevinding.

Tip

Als testers en andere partijen geografisch ver uit elkaar zitten, zoals bij outsourcing of offshoring vaak het geval is, is aan te bevelen om een web-enabled bevindingentool aan te schaffen. Hiermee kunnen alle partijen rechtstreeks de laatste stand van de bevindingenadministratie bekijken. Dit vergemakkelijkt de communicatie over bevindingen aanzienlijk.

Uitgediept

In sommige organisaties wordt de bevindingenadministratie ondergebracht in het incidentenregistratiesysteem van de productiesystemen. Hoewel dit op zich mogelijk is, bevat een dergelijk systeem veel meer informatievelden dan voor een bevinding nodig zijn. Soms kan dit aangepast worden, maar soms moeten de testers leren omgaan met het complexe systeem en niet letten op alle overbodige velden op het scherm. Dit vraagt duidelijk meer inleertijd en heeft een grotere kans op foutieve invoer van bevindingen dan een standaard bevindingenadministratie.

Wanneer de bevindingen zijn opgeslagen in een geautomatiseerde administratie, kunnen diverse rapportages gegenereerd worden. Deze zijn zeer nuttig om bepaalde trends over kwaliteit van het testobject en de voortgang van het testproces zo vroeg mogelijk te signaleren (zie paragrafen 5.3.2 en 6.3.2 “Bewaken”) . Denk hierbij bijvoorbeeld aan een constatering dat het overgrote deel van de bevindingen op (een deel van) het functioneel ontwerp betrekking heeft, of dat de bevindingen zich hoofdzakelijk op de schermafhandeling concentreren. Deze informatie kan weer gebruikt worden om tijdig bij te sturen en maatregelen te nemen.

Het succes van de bevindingenadministratie wordt in belangrijke mate bepaald door de invuldiscipline van de testers. Hiervoor moeten de testers allereerst goed weten wat elk veld inhoudt en hoe dit gevuld moet worden. Vervolgens is zeker in het begin begeleiding van en controle op het invullen van bevindingrapporten nodig. Dit is normaal gesproken een rol voor de testmanager, bevindingenbeheerder of intermediair en onderdeel van de laatste stap “Laat reviewen” in paragraaf 12.2.

De uniformiteit en consistentie van een bevindingrapport kan worden verbeterd door voor de velden de mogelijke invoerwaarden te beperken in plaats van vrije tekstvelden te gebruiken. Voorbeeld: voor de oorzaak van de bevinding kan een keuze worden toegestaan uit testbasis, testobject of testomgeving. Hiermee wordt voorkomen dat er allerlei synoniemen worden ingevoerd (‘software’, ‘code’, ‘programmatuur’, ‘programma’, ‘component’) die een latere selectie op foutoorzaak van de bevinding sterk bemoeilijken of onmogelijk maken.

Onderstaand wordt eerst beschreven wat een bevindingrapport minimaal moet bevatten. Vervolgens worden diverse aanbevelingen gegeven voor uitbreiding hiervan.

Minimale velden bevindingrapport

Een bevindingrapport bevat minimaal de volgende velden:

- *Project- of systeemnaam*
  De naam van het (test)project of van het systeem dat wordt getest.
- *Unieke identificatie bevinding*
  Een unieke identificatie, meestal in de vorm van een (volg)nummer van het bevindingrapport, ten behoeve van het beheer en de tracering van de voortgang.
- *Korte typering*
  Een korte typering van de bevinding in een beperkt aantal woorden, maximaal een zin, die bij voorkeur ook het gevolg van de bevinding duidelijk maakt. Deze typering wordt afgedrukt op bevindingenoverzichten en maakt de bevinding beter communiceerbaar.
- *Indiener*
  De naam van degene die de bevinding heeft ingediend.
- *Identificatie fase/testsoort*
  De fase of testsoort waarin de bevinding gedaan is, bijvoorbeeld ontwerp, ontwikkeling, ontwikkeltest, systeemtest, acceptatietest of implementatie.
- *Ernst*
  De door de tester voorgestelde ernstcategorie. Deze ernstcategorie geeft de schade aan de bedrijfsvoering weer. Bijvoorbeeld:
  - Productieblokkerend: Er is direct (veel) geld mee gemoeid, bijvoorbeeld omdat de bevinding de bedrijfsvoering stillegt indien het systeem in productie gaat;
  - Ernstig: Er is (minder) geld mee gemoeid, bijvoorbeeld omdat de gebruiker handmatig zaken moet herstellen of aanvullen;
  - Storend: Er is weinig of geen geld mee gemoeid, bijvoorbeeld afkappingen van alfanumerieke data op scherm of zaken met betrekking tot gebruikersvriendelijkheid;
  - Cosmetisch: Verkeerde lay-out (positie van velden, kleuren) waar de externe klant geen last van heeft maar die wel storend kan zijn voor de eigen medewerker.
- *Prioriteit*
  De door de tester voorgestelde prioriteit van oplossen. Mogelijke onderverdeling:
  - Direct herstel nodig: Bijvoorbeeld binnen 48 uur een patch beschikbaar die het probleem (voorlopig) verhelpt. Het testproces of de huidige bedrijfsvoering (als het een bevinding uit productie betreft) wordt op ernstige wijze geblokkeerd;
  - Herstel nodig binnen de huidige release. Het huidige proces kan eventueel nog met work-arounds voortgaan, maar productie mag niet met dit probleem opgescheept worden;
  - Herstel op termijn nodig, maar hoeft pas in een volgende release beschikbaar te zijn. Het probleem doet zich (voorlopig) niet voor in productie of de schade is gering.

Uitgediept

Het lijkt op het eerste gezicht niet zo belangrijk om onderscheid te maken tussen ernst en prioriteit. Normaal gesproken lopen deze ook synchroon, dus hoge ernst impliceert hoge prioriteit van oplossen. Toch gaat dit niet altijd op en dat is de reden om beide categorieën apart te onderscheiden. De volgende voorbeelden illustreren dit:

1. Bij een nieuwe release is de interne naamgeving in de programmatuur aangepast. De gebruiker ziet of merkt hier niets van maar de geautomatiseerde testsuite werkt opeens niet meer. Dit is een bevinding met lage ernst maar test-blokkerend en dus met zeer hoge prioriteit.
2. De gebruiker kan een bepaalde bevinding zo storend vinden dat die bevinding niet in productie mag optreden.

Dit is bijvoorbeeld een typefout in een brief naar een klant. Ook dit is een bevinding met lage ernst die toch herstel vóór het in productie gaan vergt.

3. Een potentieel zeer ernstige bevinding, bijvoorbeeld het crashen van de applicatie met dataverlies tot gevolg, treedt uitsluitend op in zeer specifieke omstandigheden die niet vaak voorkomen. Er is een work-around beschikbaar. De ernst is hoog, maar de prioriteit kan wegens de work-around lager gezet worden.

- *Oorzaak*
  Met de oorzaak geeft de tester aan waar de oorzaak van de fout volgens hem ligt, bijvoorbeeld: TB: testbasis (requirements, specificaties)
  PR: programmatuur
  DOC: documentatie
  TIS: technische infrastructuur.
- *Identificatie testobject*
  Het (onderdeel van het) testobject waarop de bevinding betrekking heeft, moet in deze rubriek worden aange geven. Onderdelen van het testobject kunnen bijvoorbeeld deelobjecten, functies of schermen zijn. Optioneel kan een nadere detaillering worden doorgevoerd door het veld te splitsen in meerdere velden, zodat bijvoorbeeld deelsysteem en functie kunnen worden ingevuld. Tevens wordt het versienummer of de versiedatum van het testobject vermeld.
- *Testspecificatie*
  Een verwijzing naar het testgeval waarop de bevinding betrekking heeft, zoveel mogelijk met een relatie naar de testbasis.
- *Omschrijving bevinding*
  De geconstateerde bevinding wordt conform de richtlijnen in paragraaf 12.2 zo goed mogelijk beschreven.
- *Bijlagen*
  Indien verduidelijking of bewijs noodzakelijk zijn, worden bijlagen toegevoegd. Een bijlage is van papier, zoals een schermafdruk of een overzicht, of een (verwijzing naar een) elektronisch bestand.
- *Oplosser*
  De naam van degene die de bevinding in oplossing heeft, heeft opgelost of heeft afgewezen.
- *Toelichting oplossing*
  De oplosser licht de gekozen oplossing (of reden van afwijzing) van de bevinding toe.
- *Opgelost in product*
  De identificatie van het product inclusief versienummer waarin de bevinding opgelost moet zijn.
- *Status + datum*
  De verschillende stadia van de levenscyclus van de bevinding worden geadministreerd, tot en met hertest. Dit is nodig om de bevinding te kunnen bewaken. Op z'n eenvoudigst zijn de statussen "Nieuw", "In oplossing", "Uitgesteld", "Afgewezen", "Opgelost", "In hertest" en "Afgedaan". De status wordt tevens voorzien van de datum.

Mogelijke uitbreidingen

Naast bovenstaande velden kunnen aan het bevindingrapport nog diverse andere velden toegevoegd worden. De voordelen van het opnemen van één of meer van onderstaande velden zijn beter beheer en meer inzicht in de kwaliteit en trends. De nadelen zijn de extra administratie en complexiteit. De ervaring leert dat de voordelen ruimschoots opwegen tegen de nadelen bij middelgrote en grote tests of in gevallen waarbij veel communicatie over de bevindingen tussen verschillende partijen nodig is.

- *Identificatie testomgeving*
  De gebruikte testomgeving met de identificatie van de gebruikte uitgangssituatie.
- *Identificatie testbasis*
  De gebruikte testbasis: naam van het testbasisdocument, inclusief versienummer, eventueel aangevuld met specifiek requirement- nummer.
- *Voorlopige ernst*
  Voorlopig: de door de tester voorgestelde ernstcategorie.
- *Voorlopige prioriteit*
  Voorlopig: de door de tester voorgestelde prioriteit van oplossen.
- *Voorlopige oorzaak*
  Voorlopig: de door de tester ingeschatte oorzaak van de bevinding.
- *Kwaliteitsattribuut*
  Het door de tester vastgestelde kwaliteitsattribuut, waarop de bevinding betrekking heeft.

Met betrekking tot de oplossing:

- *Definitieve ernst*
  De uiteindelijk door het bevindingenforum bepaalde ernstcategorie.
- *Definitieve prioriteit*
  De uiteindelijk door het bevindingenforum bepaalde prioriteit van oplossen.
- *Definitieve oorzaak*
  De uiteindelijk door het bevindingenforum bepaalde oorzaak van de bevinding. Naast de categorieën genoemd bij het minimale bevindingrapport komt hier nog de categorie “Testen” bij.
- *Gewenste datum of release opgelost*
  Er wordt een datum of productrelease gesteld waarop de bevinding opgelost moet zijn.

Met betrekking tot hertesten:

- *Hertester*
  De naam van de tester die de hertest uitvoert.
- *Identificatie testomgeving*
  De gebruikte testomgeving met de identificatie van de gebruikte uitgangssituatie.
- *Identificatie testbasis*
  De gebruikte testbasis: naam van het testbasisdocument, inclusief versienummer, eventueel aangevuld met specifiek requirement- nummer.
- *Identificatie testobject*
  Het (onderdeel van het) testobject dat gehertest is. Tevens wordt het versienummer of de versiedatum van het testobject vermeld.

Daarnaast kunnen aan het test-, bevindingenforum- en hertest-deel opmerkingen-velden worden toegevoegd waarmee optioneel extra informatie gegeven kan worden, bijvoorbeeld over corresponderende bevindingen of de identificatie van het wijzigingsvoorstel waarmee de afhandeling van de bevinding in een andere procedure ondergebracht wordt.

12.4 Procedure

Wanneer de bevinding eenmaal in de administratie is ingevoerd, gaat deze de bevindingenprocedure in.

De voortgang van (het oplossen van) bevindingen wordt besproken in een periodiek bevindingenoverleg. Tijdens het voorbereiden en specificeren van tests wordt dit overleg normaal gesproken één of twee keer per week gehouden. Tijdens testuitvoering loopt dit vaak op tot elke dag.

Deelnemers aan het overleg zijn vertegenwoordigers van de partijen die de bevindingen indienen en/of afhandelen. Vanuit testen is dit de testmanager, bevindingenbeheerder of intermediair. Soms wordt een tester uitgenodigd om een bevinding toe te lichten. Andere partijen kunnen zijn de gebruikersorganisatie, functioneel beheer, systeemontwikkeling en systeembeheer. Het bevindingenoverleg wordt ook wel samengevoegd met de behandeling van wijzigingsvoorstellen in bijvoorbeeld een Change Control Board.

Tips

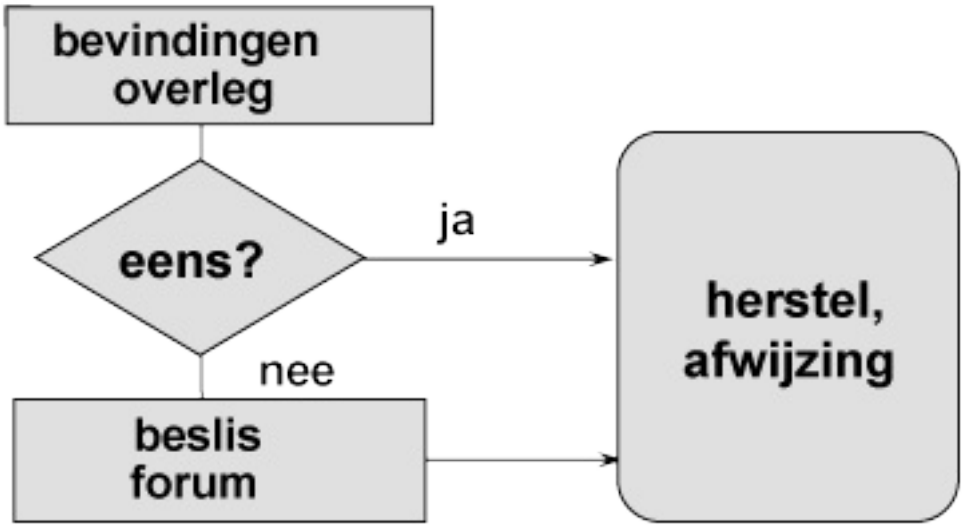
- Conference call
  Wanneer de partijen over verschillende locaties (over de wereld) verspreid zijn, is dit geen reden om geen bevindingenoverleg te voeren. Conference calls of video conferencing bieden dan uitkomst.
- Zorg dat iedere deelnemer goed op de hoogte is van de werkwijze van de bevindingenprocedure en wat ieders taken en verantwoordelijkheden zijn. Wie werkt bijvoorbeeld de status van de bevindingen bij na het bevindingenoverleg?

In volgorde van prioriteit overleggen de deelnemers over elke nieuwe bevinding of de bevinding opgelost moet worden en zo ja, wie dit gaan doen. In dit overleg worden de correctheid, oorzaak, prioriteit en ernst van de bevindingen én de kosten van oplossen bediscussieerd. Een bekende humoristische reactie van ontwikkelaars in dit verband is “it’s a feature, not a bug”. De vertegenwoordiger van het testen heeft tot taak om te zorgen dat het belang van een bevinding (ernst en prioriteit) voldoende duidelijk wordt bij alle partijen. Het overleg kan ook aan de indiener van de bevinding vragen om aanvullende informatie te geven of verder onderzoek te doen. De deelnemers aan het overleg bepalen na het voeren van

de benodigde discussies de definitieve waarden voor oorzaak, prioriteit en ernst van een bevinding.

Als het bevindingenoverleg het eens wordt dat het een valide bevinding is en de kosten van oplossen acceptabel zijn, wordt de bevinding toegewezen aan een oplosser. Is het overleg het eens dat het geen valide bevinding is óf dat de kosten, doorlooptijd of regressierisico voor oplossen te hoog zijn, dan wordt deze afgewezen. Een valide bevinding die toch afgewezen wordt, staat ook wel bekend als ‘known error’. Bij afwijzing kan eventueel gekozen worden om de bevinding via een ander kanaal als formeel wijzigingsvoorstel in te dienen of een procedurele oplossing te verzinnen. Voorbeelden van procedurele oplossingen zijn toelichtingen in de helptekst, instructie van de helpdeskmedewerkers of aanpassing van de AO-procedures. Wordt het overleg het niet eens, dan wordt de bevinding geëscaleerd en doorgespeeld naar het beslisforum. Hierin zitten beslissingsbevoegde vertegenwoordigers van de partijen, zoals opdrachtgever en projectmanager, die een besluit nemen over het wel of niet (en wanneer) oplossen van de bevinding. Het beslisforum hoeft geen zelfstandig overleg te zijn, maar is vaak het projectmanagementoverleg of het stuurgroepoverleg.

Figuur 12.1 geeft de relatie tussen bevindingenoverleg en beslisforum weer:



Figuur 12.1: Bevindingenprocedure.

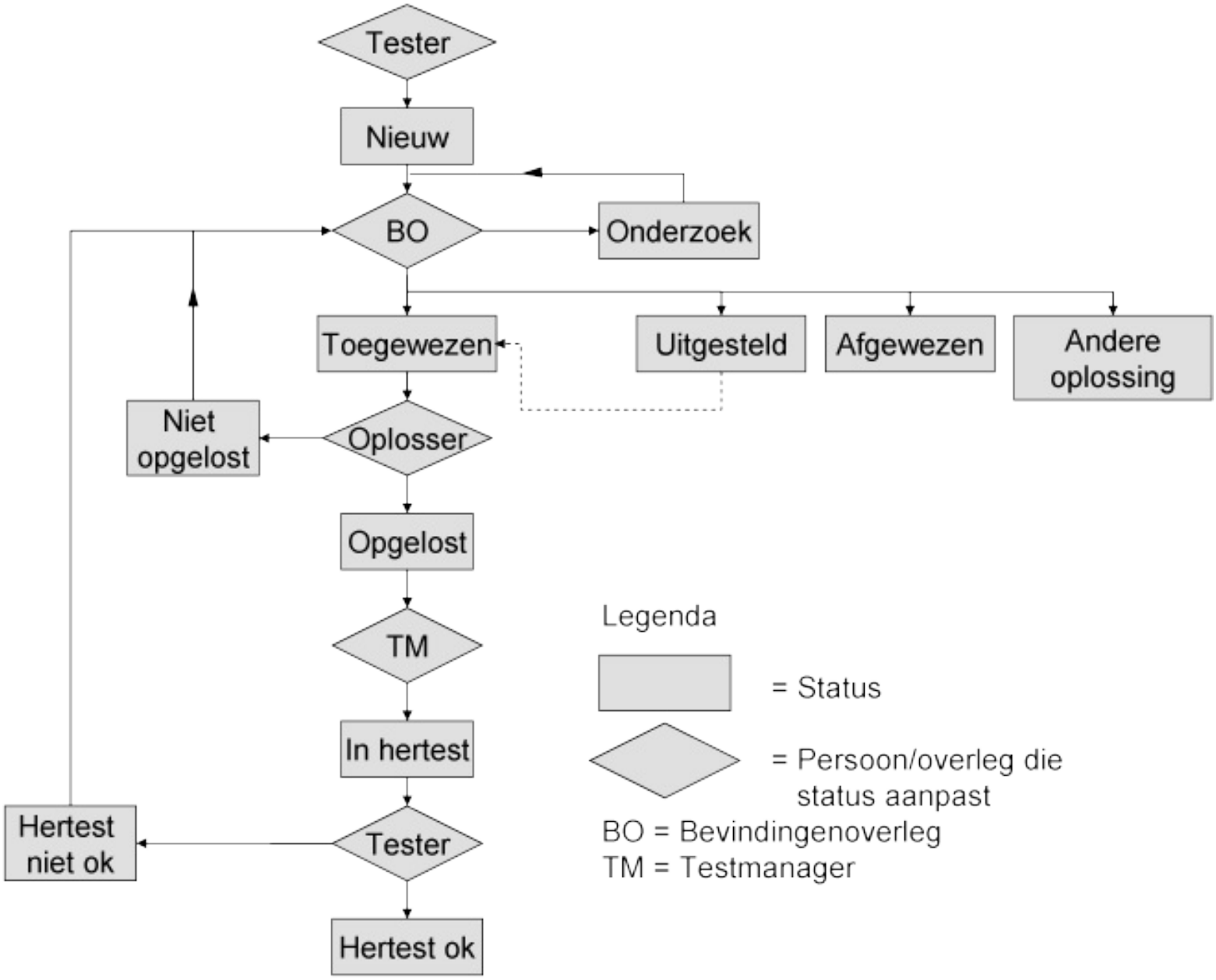
De oplosser onderzoekt de bevinding en lost deze op. Hier kan ook nog blijken dat de bevinding onterecht is (een testfout) of door een andere oplosser behandeld moet worden. In die laatste gevallen gaat de bevinding terug naar het overleg. Is de bevinding opgelost, dan kan deze op enig moment naar de testomgeving worden overgedragen om ge(her)test te worden. De tester, bij voorkeur de oorspronkelijke indiener, voert de test uit en controleert of de bevinding is opgelost. Als dat zo is, wordt de bevinding afgesloten. Blijkt de bevinding niet (goed) opgelost, dan wordt de status van de bevinding teruggezet en gaat deze opnieuw de bevindingenprocedure door.

Het hertesten van de bevinding is een essentiële stap om de bevinding af te kunnen sluiten. Het mag niet zo zijn dat de oplosser de bevinding oplost, zelf test en dan de bevinding afsluit. Het controleren of de bevinding is verholpen is de taak van de indiener (of een vervanger).

Uitgediept

De tijd die nodig is voor het uitzoeken, indienen, behandelen, oplossen en hertesten van een bevinding is aanzienlijk. Alleen al aan puur administratieve en beheerwerkzaamheden kost de gemiddelde bevinding tussen de 1 en 2 uur. Dit is een belangrijke reden om te eisen dat het testobject met voldoende kwaliteit een test ingaat. De pretest heeft tot taak om deze testbaarheid te controleren (zie paragraaf 6.7.1 “Intake testobject”).

Figuur 12.2 toont de levenscyclus van een bevinding volgens bovenstaande procedure, waarbij de teksten in de rechthoeken de status van de bevinding duiden. De ruiten staan voor de actienemers. Met de stippelpijl van “Uitgesteld” naar “Toegewezen” wordt bedoeld dat de bevinding in de huidige release wordt uitgesteld, maar in een toekomstige release alsnog opgelost moet worden.



Figuur 12.2: Levenscyclus van een bevinding.

13 Metrics

13.1 Inleiding

Het komt vaak voor dat de testmanager antwoord moet geven op een aantal vragen zoals:

- Waarom duurt het testen toch zo lang?
- Waarom is het testproces nog niet afgerond?
- Hoeveel bevindingen kan ik in productie nog verwachten?
- Hoeveel hertesten zijn er nog nodig?
- Wanneer kan er gestopt worden met testen?
- Wanneer begint het testteam met de testuitvoering?
- Vertel me maar eens wat jullie eigenlijk aan het doen zijn!
- Hoe staat het met de kwaliteit van het te testen systeem?
- Wanneer kan ik in productie?
- Hoe komt het dat het vorige testproject veel sneller verliep?
- Wat hebben jullie nou precies getest?
- Hoeveel bevindingen zijn er, en wat is de status hiervan?

Het met feiten onderbouwd beantwoorden van dit soort vragen is niet eenvoudig. De meeste vragen kunnen beantwoord worden aan de hand van een periodieke rapportage zoals deze beschreven is in paragraaf 6.3.3 “Rapporteren”. Zo’n rapportage kan alleen tot stand komen door het correct bijhouden van relevante gegevens. Deze gegevens worden herleid tot informatie. Vervolgens wordt die informatie gebruikt om een antwoord te geven op de bovenstaande vragen, ofwel Meten = Weten en Gissen = Missen.

Voor het testproces zijn metrics over de kwaliteit van het testobject en de voortgang van het testproces van groot belang. Ze worden gebruikt om het testproces te beheersen, om de testadviezen te onderbouwen en ook om systemen of testprocessen met elkaar te vergelijken. Voor het verbeteren van het testproces zijn metrics van belang om de gevolgen van bepaalde verbetermaatregelen te beoordelen, door gegevens vóór en ná het nemen van de maatregel met elkaar te vergelijken.

Het komt samengevat hierop neer: een testmanager moet een aantal zaken bijhouden om gefundeerd een oordeel te kunnen geven over de kwaliteit van het te testen object én over de kwaliteit van het testproces zelf. De volgende paragrafen beschrijven een gestructureerde aanpak om tot een set van testmetrics te komen.

13.2 GQM-Methode in zes stappen

Er zijn verschillende manieren om te komen tot een bepaalde metricsset. De meest gehanteerde vorm is de Goal-Question-Metric (GQM) methode [Basili 1994]. Dit is een top-down methode, waarbij één of meer doelstellingen geformuleerd worden. Bijvoorbeeld: welke informatie moet ik verzamelen om de vragen uit de inleiding te beantwoorden. Bij deze doelstellingen worden vragen gesteld die de basis vormen voor de metrics. De verzamelde metrics moeten antwoord geven op de gestelde vragen. De antwoorden geven onder meer aan of het doel wel of niet bereikt is. De hieronder beschreven samenvatting van de GQM-methode richt zich met name op het testaspect. In een zestal stappen wordt het GQM-proces beschreven. Dit betreft een beknopte beschrijving waarin alleen de voor de testmanager relevante zaken zijn opgenomen. Voor een uitgebreide beschrijving wordt verwezen naar de eerder genoemde GQM-literatuur.

Stap 1: Definiëren van doelen

Meten puur om te meten heeft geen enkele zin. Er moeten vooraf duidelijke en realistische doelen gesteld worden. We onderkennen hierbij twee soorten doelen:

- Kennisdoelstellingen (“weten waar we nu staan”). Deze doelen worden uitgedrukt door het gebruik van woorden als evalueer, voorspel of monitor. Te denken valt aan “Evalueer hoeveel uur er daadwerkelijk wordt besteed aan hertesten” of “Monitor de testcoverage”. Het gaat hierbij om het verkrijgen van inzicht.
- Verbeterdoelstellingen (“waar willen we naar toe”). Deze worden uitgedrukt door het gebruik van woorden als

verhoog, verlaag, verminder, verbeter of bereik. Het stellen van zulke doelen betekent dat men op de hoogte is van het feit dat er tekortkomingen zijn in het huidige testproces of in de huidige omgeving en dat men deze wil verbeteren.

Een voorbeeld van een verbeterdoelstelling is dat men binnen 18 maanden 20% besparing op het aantal testuren wil bereiken bij een gelijkblijvende testcoverage.

Om dit na te gaan dient men als eerste de volgende twee kennisdoelstellingen op te nemen:

- Inzicht in de totale testuren per project.
- Inzicht in de gehaalde testcoverage per project.

Het is van belang om na te gaan of de doelen en de (test-)volwassenheid van de organisatie met elkaar overeenstemmen. Het heeft geen zin om als doel te stellen dat er bepaalde coverage gehaald moet worden op programmaregels als hiervoor niet de benodigde capaciteit (kennis, tijd, tools) beschikbaar is.

Voorbeeld: Weten waar we nu staan

Stap 2: Stellen van vragen per doel

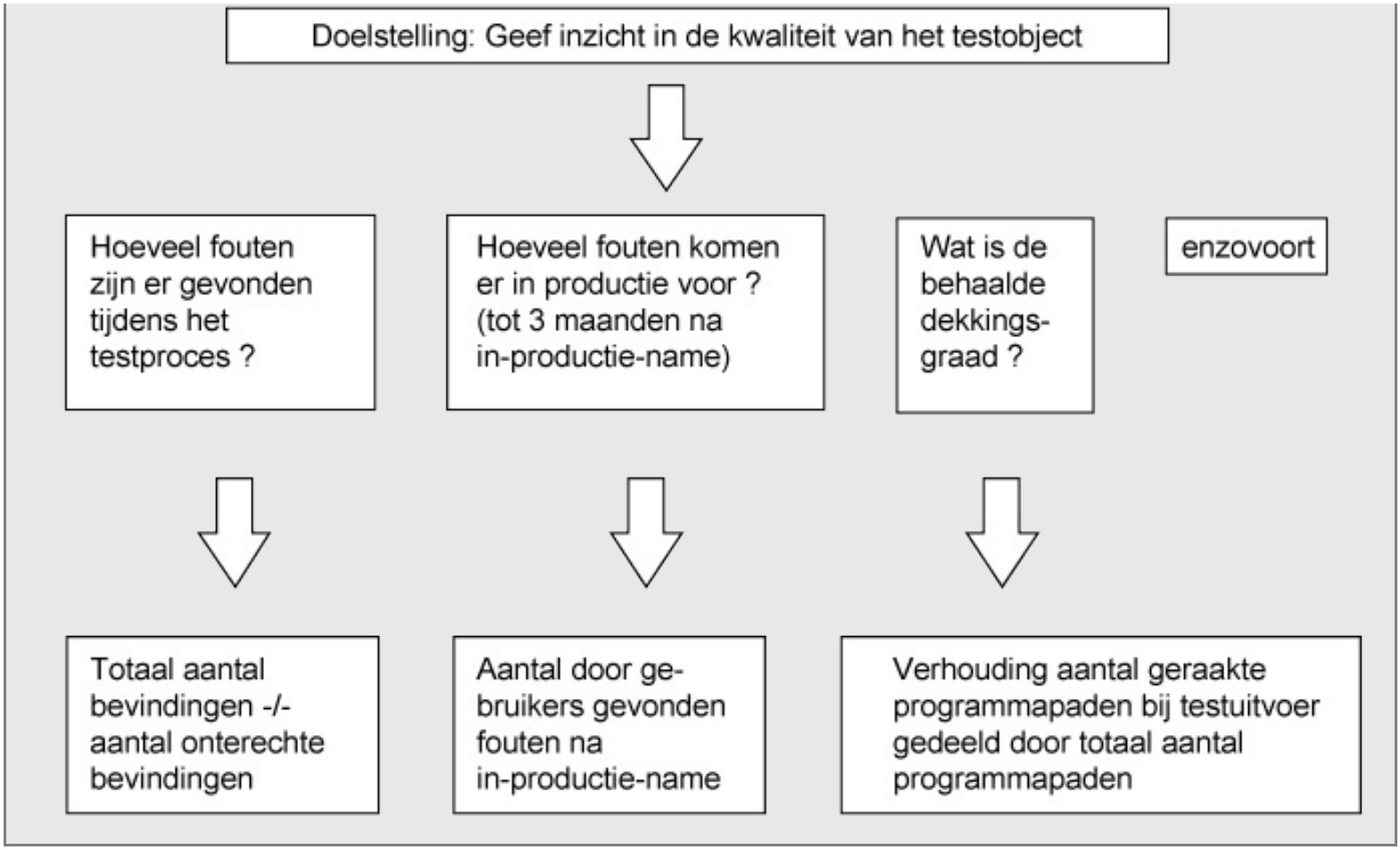
Per doel worden meerdere vragen gesteld. De vragen worden op zodanige manier geformuleerd dat ze fungeren als specificatie van een metric. Daarnaast is het zo dat per vraag duidelijk gesteld kan worden wie verantwoordelijk is voor de daarvoor aangeleverde testmetrics. Vanuit bovenstaande doelstelling zijn diverse vragen af te leiden. We beperken het aantal vragen in dit voorbeeld tot drie.

Voorbeeld

Stap 3: Van vragen naar metrics

Vanuit de gestelde vragen worden de bijbehorende metrics afgeleid. Deze metrics vormen tezamen de totale metricsset. Deze metrics worden tijdens het testproces verzameld.

Voorbeeld



Door de juiste vragen te stellen komt men ‘automatisch’ tot een juiste set van metrics voor een bepaalde doelstelling. Het is van belang iedere metric goed te definiëren en te specificeren. Wat is bijvoorbeeld een bevinding?

Stap 4: Datacollectie en analyse

Gedurende het lopende testtraject worden diverse gegevens bijgehouden.

Een manier om de zaak eenvoudig te houden is het gebruik van formulieren/ templates (indien mogelijk in geautomatiseerde vorm). De gegevens moeten compleet en gemakkelijk te interpreteren zijn. Bij het ontwerpen van deze formulieren dient men op de volgende punten te letten:

- Welke metrics worden verzameld op hetzelfde formulier.
- Validatie: hoe gemakkelijk is het om te controleren of de data compleet en juist zijn.
- Traceerbaarheid: formulieren voorzien van datum, project-id, configuratie management data, verzamelaar data, enzovoort. Houdt er rekening mee dat deze gegevens soms voor lange tijd bewaard moeten kunnen worden.
- Electronische verwerkbaarheid.

Zodra de gegevens verzameld zijn, moet er gestart worden met de analyse hiervan. Op dit moment is het nog mogelijk om zaken te corrigeren. Als men hier te lang mee wacht, wordt de kans om de gegevens te herstellen steeds kleiner. Hier kan men bijvoorbeeld denken aan het boeken van uren op de verkeerde activiteitencode.

Stap 5: Presentatie en distributie van de meetgegevens

Zowel bij de testrapporten over de kwaliteit van het te testen product als testrapporten over het testproces, wordt gebruik gemaakt van de verzamelde meetgegevens. Een goede terugkoppeling is tevens van belang in het kader van de motivatie van de betrokkenen en de validatie van de meetgegevens.

Stap 6: Relateren van de meetgegevens aan de vragen en doelstellingen

In deze laatste stap wordt bekeken in hoeverre de kengetallen (antwoorden op de gestelde vragen) voldoende inzicht geven op de vraag of de gestelde doelen bereikt zijn. Het is mogelijk dat deze situatie een beginsituatie is voor een nieuwe GQM-cyclus. Op deze manier is men continu bezig het testproces te verbeteren.

13.3 Hints en Tips

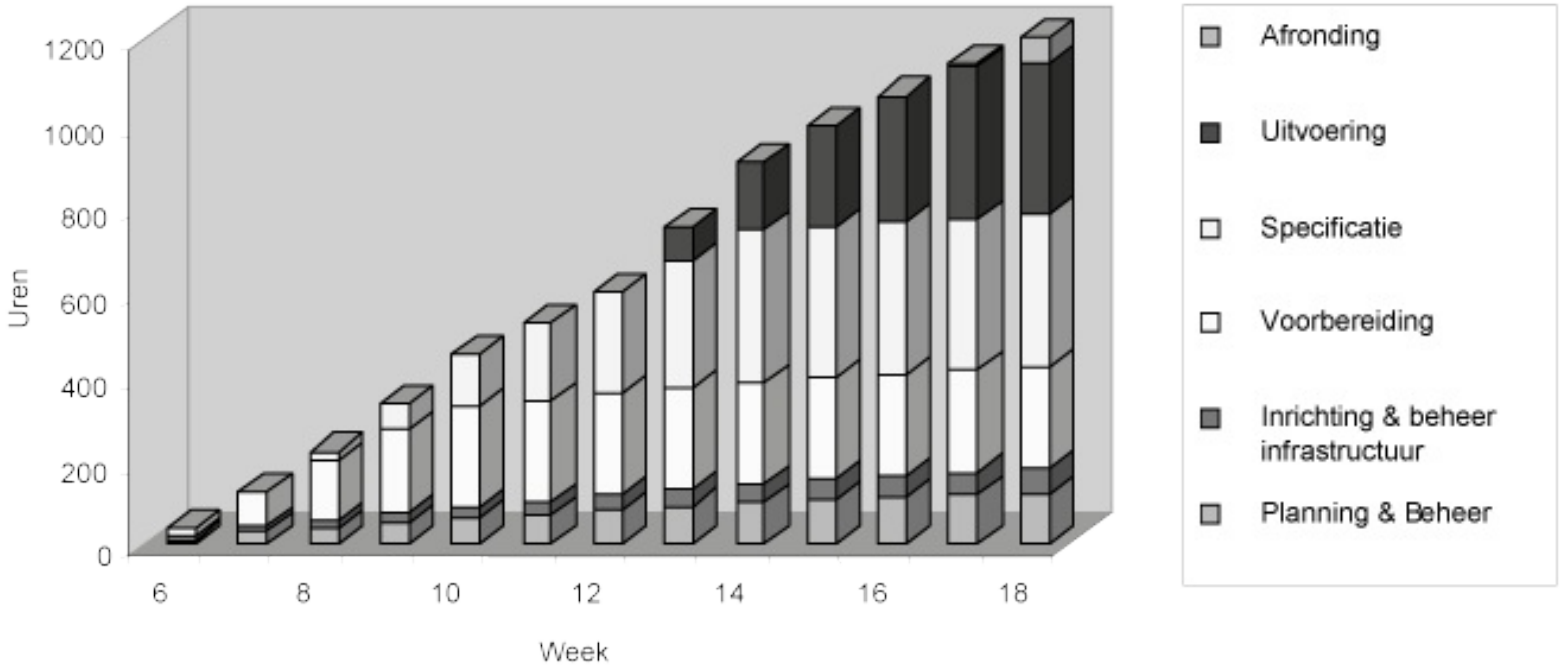
Bij het verzamelen van metrics moet de testmanager met de volgende zaken rekening houden:

- Start met een beperkte metricsset en bouw deze langzaam op.
- Houd de metrics eenvoudig. De definitie moet aansluiten bij de intuïtie van de betrokkenen. Wees bijvoorbeeld behoudend in het gebruik van diverse formules. Hoe ingewikkelder de formules zijn, hoe moeilijker het is om deze te interpreteren.
- Kies metrics die relatief eenvoudig te verzamelen zijn en die gemakkelijk geaccepteerd worden. Hoe lastiger het is om gegevens te verzamelen, hoe groter de kans is dat het niet aanslaat.
- Verzamel de gegevens zoveel mogelijk op geautomatiseerde wijze. Op deze manier gaat de datacollectie het snelst en worden geen handmatige fouten geïntroduceerd in de betreffende dataset.
- Blijf letten op de motivatie van de testers om accuraat te noteren. Bijvoorbeeld bij urenregistraties komt het wel eens voor dat op de verkeerde (lees: nog niet volgeboekte) codes geschreven wordt.
- Vermijd bij de presentaties gecompliceerde statistische technieken en modellen. Laat de presentatievorm afhankelijk zijn van de te presenteren data (tabellen, grafieken, cirkeldiagrammen, enzovoort).
- Koppel zo snel mogelijk terug naar de testers. Laat hen zien wat je met de informatie doet.

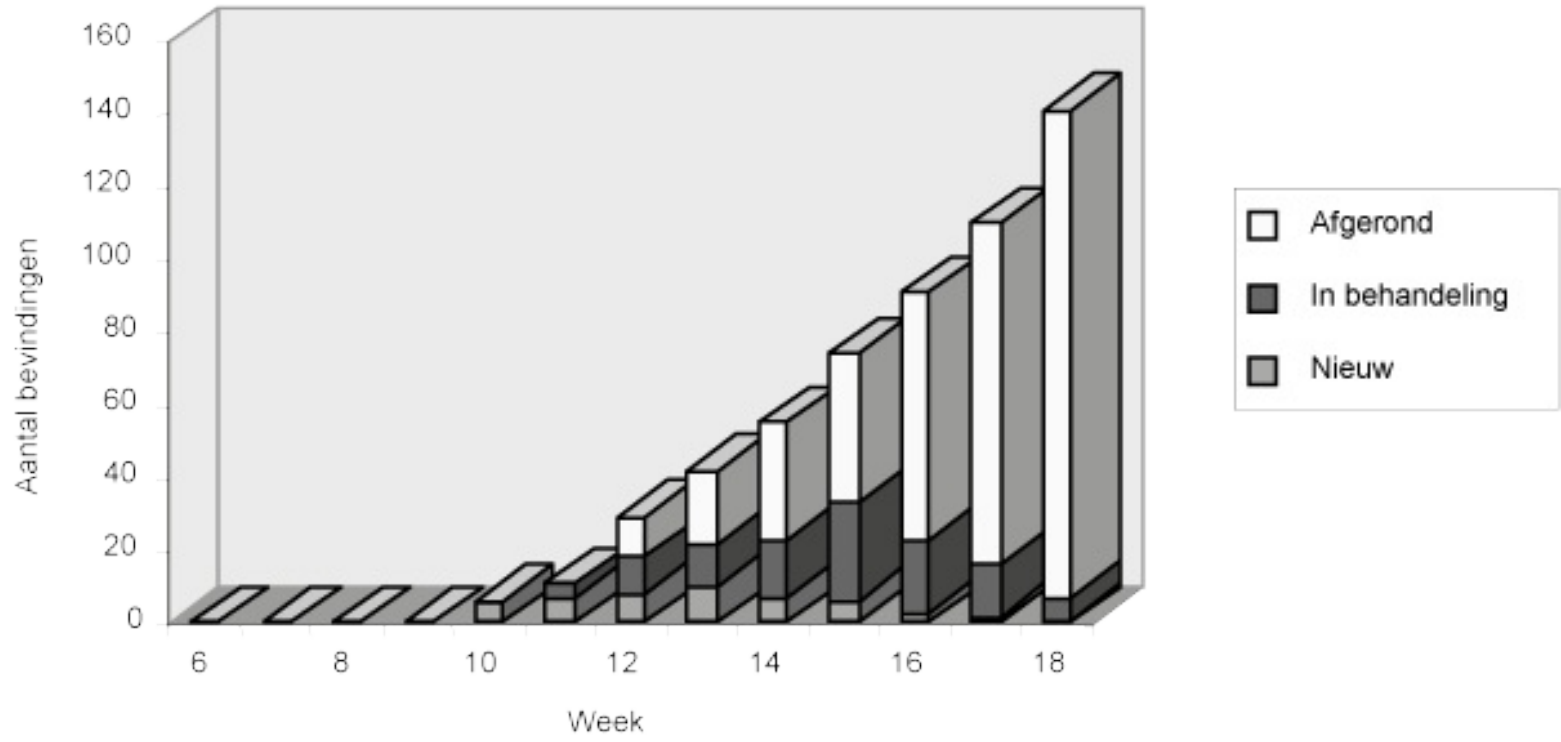
13.4 Praktische beginset testmetrics

Voor de testmanager die begint met een metrics-programma wordt hieronder aangegeven waar het best mee begonnen kan worden. De metricsset die beschreven staat is een beginset, die met weinig kosten en moeite in de praktijk werkbaar is. In paragraaf 6.3.3 “Rapporteren” zijn een aantal specifiekere teststatistieken en voortgangsrapportages vermeld.

- Het bijhouden van uren met behulp van urencodes. Houdt per tester de volgende zaken bij: datum, project, TMap-fase, activiteit, aantal bestede uren. Het is aan te bevelen om naast deze velden ook een veld ‘opmerkingen’ op te nemen, zodat gecontroleerd kan worden of de gegevens goed ingevuld zijn. Door op deze manier de uren bij te houden is het mogelijk om een duidelijk inzicht te krijgen in de bestede uren per TMap-fase (zie figuur 13.1). Ook de opdrachtgever kan op deze wijze inzicht verkrijgen in de voortgang van het testproces. Het is aan te bevelen dit soort urenrapportages wekelijks te doen voor projecten die maximaal 3 tot 4 maanden duren. Voor projecten tot ruim een half jaar kan deze 2-wekelijks. Voor projecten die langer dan een jaar duren, kan het best maandelijks of 4-wekelijks gerapporteerd worden.
- Breng de testproducten (testplannen, testscripts en dergelijke), de testbasis en het testobject in kaart. Houdt de volgende zaken bij: documentnaam, opleverdatum, TMap-fase bij oplevering, versie en een kenmerk dat iets zegt over de kwantiteit. Hier kan bijvoorbeeld gedacht worden aan het aantal testgevallen voor de testscripts of het aantal pagina’s voor de overige documenten. Bij de testbasis kan men het aantal user-requirements als kwantiteitskenmerk opnemen.
- Rapporteer over de voortgang van de bevindingen. In hoofdstuk 12 “Bevindingenbeheer” is beschreven hoe een bevindingenadministratie ingericht kan worden. Een voorbeeld van deze rapportagevorm is hierna weergegeven (zie figuur 13.2):



Figuur 13.1: Voorbeeld bestede uren testproces per TMap-fase.



Figuur 13.2: Voorbeeld voortgangsoverzicht bevindingen.

Met deze elementaire metrics (uren, documenten en bevindingen) kan een uitspraak gedaan worden over de productiviteit van het testproces. Hierbij dient men zich er van bewust te zijn dat deze productiviteit in relatie gezien moet worden met de benodigde inspanningen en omvang van het testproject. Voorbeeld: in de eerste 10 uur testtijd kan men meer fouten per uur vinden dan in 400 uur testtijd, gewoonweg omdat de eerste fouten sneller gevonden worden dan de laatste.

Vanuit deze elementaire set zijn met betrekking tot productiviteit de volgende metrics af te leiden:

- aantal bevindingen per uur (en per uur testuitvoering)
- aantal uitgevoerde testgevallen per uur
- aantal gespecificeerde testscripts per uur (en per uur testspecificatie)
- aantal bevindingen per testscript verhouding tussen uren volgens TMap-fasering.

De volgende metrics zijn te bepalen als het aantal functiepunten of het aantal ‘kilo lines of code’ (KLOC) van het te testen object bekend is:

- aantal testuren per functiepunt (of KLOC)
- aantal bevindingen per functiepunt (of KLOC)
- aantal testgevallen per functiepunt (of KLOC).

Bij de testbasis kunnen de volgende metrics worden bepaald:

- aantal testuren per pagina testbasis
- aantal bevindingen per pagina testbasis
- aantal testgevallen per pagina testbasis
- aantal pagina's testbasis per functiepunt.

Als ook bekend is hoeveel fouten er de eerste drie maanden in productie optreden, is de volgende metric te bepalen:

- Foutvindeffectiviteit van een testsoort: aantal testbevindingen in een testsoort gedeeld door het totaal aantal aanwezige bevindingen. Deze metric wordt ook wel Defect Detection Percentage (DDP) genoemd. Bij het berekenen van de DDP gelden volgende aannames:
  – alle bevindingen worden meegeteld
  – ernstklasse van de bevindingen wordt niet gewogen meegenomen
  – na de eerste drie maanden dat het systeem in productie is, bevinden zich er vrijwel geen fouten meer in het systeem.

De DDP kan zowel per testsoort als 'overall' worden berekend. De DDP per testsoort wordt berekend door het aantal gevonden fouten van desbetreffende testsoort te delen door de som van dit aantal gevonden fouten én het aantal gevonden fouten uit de volgende testsoort(en) én/óf de eerste drie maanden productie. De 'overall' DDP wordt berekend door het totaal aantal gevonden fouten (van alle testsoorten samen) te delen door de som van dit aantal gevonden fouten én de gevonden fouten uit de eerste drie maanden productie.

Voorbeeld

DDP berekeningen

Testsoort	Gevonden fouten
Systeemtest (ST)	100
Acceptatietest (AT)	60
3 Maanden productie	40

DDP ST (nadat de AT is uitgevoerd) : (100 / 100+60) = 63%
DDP ST (na 3 maanden productie) : (100 / 100+60+40) = 50%
DDP AT (na 3 maanden productie) : (60 / 60+40) = 60%
DDP 'overall' : (100+60 / 100+60+40) = 80%

Enkele oorzaken voor een hoog of laag DDP kunnen zijn:

- Hoog DDP
  – de tests zijn erg goed uitgevoerd
  – het systeem is nog niet veel gebruikt
  – de volgende testsoort is niet goed uitgevoerd.
- Laag DDP
  – de tests zijn niet goed uitgevoerd
  – de testbasis was niet goed en daardoor de daarvan afgeleide tests ook niet
  – de kwaliteit van het testobject is niet goed (bevat teveel fouten om tijdens de beschikbare testtijd te vinden)
  – de testdoorlooptijd is ingekort.

Door bovenstaande metrics bij te houden, aangevuld met hier en daar wat specifieke zaken, komt men tot onderstaande lijst met metrics.

13.5 Metricslijst

In onderstaande, niet uitputtende, lijst worden een aantal vaak gebruikte metrics genoemd. Deze metrics zijn mogelijk te

hanteren graadmeters om een uitspraak te doen over de kwaliteit van het te testen object of om de kwaliteit van het testproces te meten en te vergelijken met de door de organisatie gestelde norm. Alle metrics kunnen vanzelfsprekend ook worden gebruikt bij de rapportage aan de opdrachtgever:

- *Aantal gevonden fouten*
  Verhouding tussen het aantal gevonden fouten en de grootte van het systeem per tijdseenheid testen.
- *Uitgevoerde instructies*
  Verhouding tussen het aantal geteste programma-instructies en het totaal aantal programma-instructies. Er zijn tools beschikbaar die een dergelijke metric opleveren.
- *Aantal tests*
  Verhouding tussen het aantal tests en de systeemgrootte (bijvoorbeeld uitgedrukt in functiepunten). Dit geeft aan hoeveel tests er nodig zijn om een onderdeel te testen.
- *Aantal geteste paden*
  Verhouding tussen de geteste en het totaal aantal aanwezige logische paden.
- *Aantal fouten tijdens productie*
  Geeft een indicatie van het aantal tijdens het testproces niet gevonden fouten.
- *Foutvindeffectiviteit*
  Het totaal aantal gevonden fouten tijdens het testen gedeeld door het totaal aantal, mede tijdens productie te bepalen, fouten.
- *Testkosten*
  Verhouding tussen de testkosten en de totale ontwikkelkosten. Een definitie van de verschillende kosten vooraf is hierbij essentieel.
- *Kosten per gedetecteerde fout*
  Totale testkosten gedeeld door het aantal gevonden fouten.
- *Budgetuitputting*
  Verhouding tussen het budget en de werkelijke testkosten.
- *Testefficiëntie*
  Het aantal benodigde tests versus het aantal gevonden fouten.
- *Automatiseringsgraad van het testen*
  Verhouding tussen het aantal handmatig uitgevoerde en het aantal geautomatiseerd uitgevoerde tests.
- *Aantal gevonden fouten (relatief)*
  Verhouding tussen het aantal gevonden fouten en de grootte van het systeem (in functiepunten of KLOC) per tijdseenheid testen.
- *Problemen als gevolg van niet geteste aanpassingen*
  Problemen door niet geteste aanpassingen, als onderdeel van het totaal aantal problemen, ontstaan door wijzigingen.
- *Problemen na geteste aanpassingen*
- Problemen door geteste aanpassingen, als onderdeel van het totaal aantal problemen, ontstaan door wijzigingen.
- *Besparing van de test*
  Geeft aan hoeveel er is bespaard door de test uit te voeren. Met andere woorden: hoe groot zou het verlies zijn, als de test niet was uitgevoerd?

14 Testontwerptechnieken

14.1 Inleiding

Waarom testontwerptechnieken?

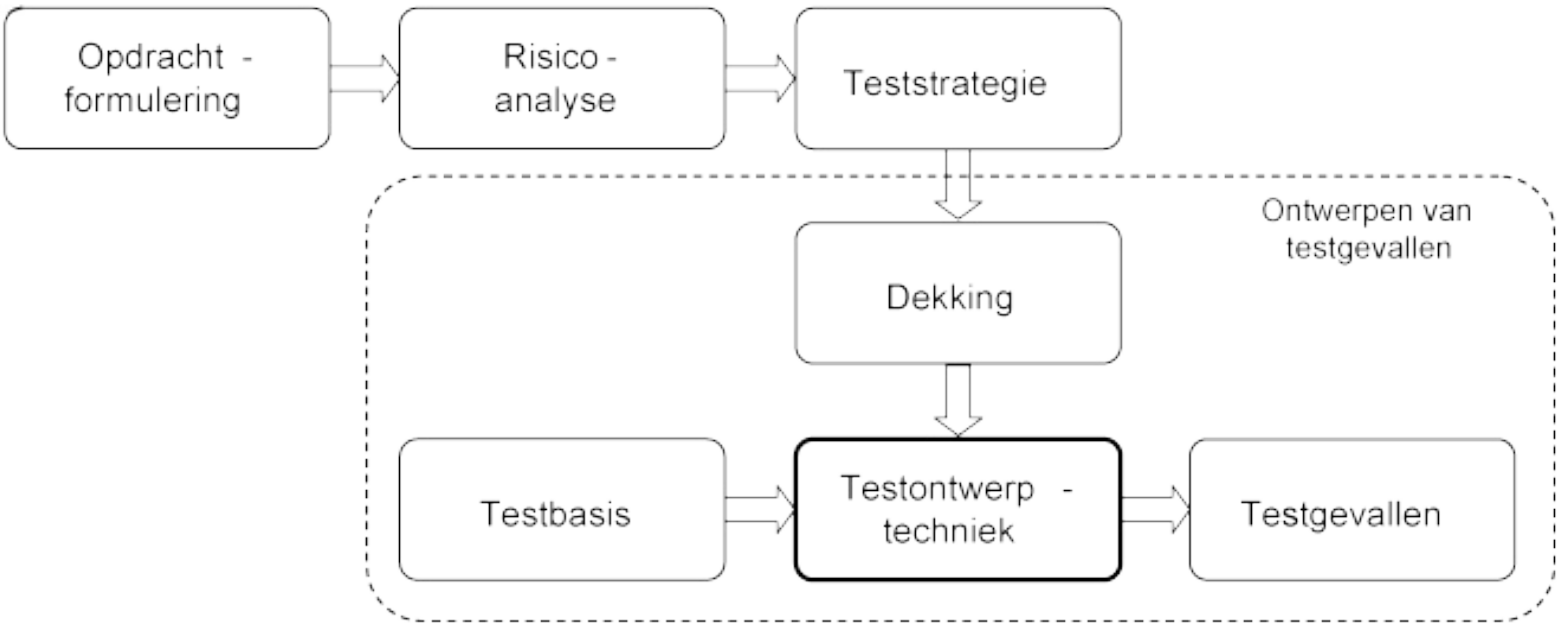
Het doel van testen is het adviseren over kwaliteit en risico's. Om dit te kunnen doen, moet de tester informatie verzamelen over het systeemgedrag. Het belangrijkste middel hiervoor is het uitvoeren van testgevallen. De grote vragen zijn nu: "Welke testgevallen? Hoeveel? En hoe komen we aan die testgevallen?" Bij het beantwoorden van deze vragen spelen testontwerptechnieken een belangrijke rol.

Definitie

Een testontwerptechniek is een gestandaardiseerde werkwijze om vanuit een bepaalde testbasis testgevallen af te leiden die een bepaalde dekking bereiken.

Het ontwerpen van testgevallen is de essentiële schakel tussen de teststrategie en de concrete testgevallen waarmee deze teststrategie ten uitvoer wordt gebracht. Dit vindt plaats in de context van testopdracht tot testgevallen (zie ook paragraaf 6.2 "Fase Planning") waarvan de rode draad als volgt geschetst kan worden (zie figuur 14.1):

- Binnen de randvoorwaarden van tijd en kosten die in de *opdrachtformulering* gesteld zijn, is het niet mogelijk om alles te testen. Er moeten keuzes gemaakt worden hoe ver men wil gaan met testen.
- Hoe belangrijker iets is, hoe zwaarder het getest moet worden. Iets is erg belangrijk als het falen ervan een grote schade betekent voor de business of opdrachtgever. Dit wordt in kaart gebracht met een *risicoanalyse*.
- Door middel van een *teststrategie* wordt een overzicht gemaakt van welke onderdelen hoe zwaar getest gaan worden, zodanig dat de eerder benoemde risico's zo adequaat mogelijk afgedekt worden.
- De beslissingen over zwaar en minder zwaar testen worden vertaald naar concrete uitspraken over welke *dekking* nagestreefd wordt.
- Onder meer afhankelijk van de beschikbare *testbasis* worden geschikte *testontwerptechnieken* gekozen om deze dekking te realiseren.
- Het toepassen van deze technieken leidt uiteindelijk tot de set *testgevallen* die nodig is om de testopdracht goed uit te voeren.



Figuur 14.1: Testontwerptechnieken binnen de context van "opdrachtformulering" tot "testgevallen".

Belang van testontwerptechnieken

Hieronder worden diverse argumenten gegeven die het belang en de voordelen aangeven van het toepassen van testontwerptechnieken en het vastleggen ervan in de testspecificaties. De argumenten zijn:

- Het geeft een onderbouwde invulling van de teststrategie: de afgesproken dekking op de afgesproken plaats;
- Omdat een testontwerptechniek gericht is op bereiken van een bepaalde dekking voor het vinden van bepaalde typen fouten (bijvoorbeeld in de interfaces, in de invoercontroles of in de verwerking), worden dergelijke fouten op een effectievere wijze opgespoord dan door ad hoc testgevallen te specificeren;
- De tests zijn reproduceerbaar, omdat de volgorde en de inhoud van de testuitvoering in detail beschreven zijn;
- De gestandaardiseerde werkwijze maakt het testproces onafhankelijk van de persoon die de testgevallen specificeert en uitvoert;
- De gestandaardiseerde werkwijze maakt de testspecificaties overdraagbaar en onderhoudbaar;
- Het testproces is beter planbaar en beheersbaar, omdat de processen van testspecificatie en -uitvoering in goed gedefinieerde blokken kunnen worden opgedeeld.

Leeswijzer

Het vervolg van dit hoofdstuk is als volgt opgebouwd:

Paragraaf 14.2 gaat dieper in op enkele essentiële begrippen rondom testontwerptechnieken. De begrippen “testsituatie”, “testgeval”, “testscript” en hun samenhang worden toegelicht. Vervolgens wordt het fenomeen “dekking” grondig besproken en wordt uitgelegd waarom er sprake is van meerdere soorten dekking en daarmee de behoefte aan meerdere testontwerptechnieken.

Vervolgens wordt in paragraaf 14.3 een overzicht gegeven van de belangrijkste dekkingsvormen, gerelateerd aan het soort informatie dat in de testbasis ter beschikking staat. Bijbehorende basistechnieken worden uitgelegd om de betreffende dekking te behalen.

Tenslotte wordt in paragraaf 14.4 een aantal testontwerptechnieken uitgelegd en met voorbeeld uitgewerkt. Daar waar een testontwerptechniek één of meer basistechnieken toepast, worden deze niet opnieuw uitgelegd maar wordt verwezen naar paragraaf 14.3. Het is een gevarieerde set testontwerptechieken waarmee de tester zeer veel praktijksituaties goed aan kan.

De lezer met een puur pragmatische insteek die alleen geïnteresseerd is in het leren van enkele specifieke testontwerptechnieken, kan zich in eerste instantie beperken tot paragraaf 14.4. Voor uitleg over de daarbij benodigde basistechnieken wordt hij automatisch doorverwezen naar paragraaf 14.3.

De lezer die tevens geïnteresseerd is in onderliggende principes van testontwerp en in het bredere scala aan mogelijkheden voor het ontwerpen van testgevallen, zal zeker ook de paragrafen 14.2 en 14.3 willen bestuderen.

14.2 Essentiële begrippen rondom testontwerp

14.2.1 Testsituatie, testgeval en testscript

In het ideale geval zouden we dankzij het testen de zekerheid hebben dat het systeem *onder alle omstandigheden* het juiste of gewenste gedrag vertoont. In werkelijkheid kunnen niet alle omstandigheden getest worden, maar slechts een subset die een direct gevolg is van de beslissingen en keuzes in het testontwerp.

Het testontwerp leidt tot een hiërarchische indeling van testsituaties, testgevallen en testscripts. Deze begrippen worden verderop in deze paragraaf uitgebreid besproken. De relatie tussen de begrippen is weergegeven in figuur 14.2 en kan worden samengevat als:

- Iedere testsituatie komt in minimaal één testgeval voor.
- Een logisch testgeval dekt één of meer testsituaties af.
- Elk logisch testgeval wordt concreet uitgewerkt in één fysiek testgeval.
- Elk fysiek testgeval komt in één testscript voor.

Figuur 14.2: Relaties tussen de begrippen testsituaties – testgevallen – testscripts.

De figuur laat ook het onderscheid zien tussen het logische en het fysieke deel van het testontwerp:

- Het *logisch testontwerp* bestaat uit de testsituaties en de logische testgevallen.
- Het is het deel dat aantoont dat de gewenste dekking gehaald wordt en dat daarmee aan de teststrategie voldaan wordt.
- Het *fysiek testontwerp* bestaat uit de concreet uitgewerkte fysieke testgevallen, vastgelegd in testscripts. Dit zorgt voor een gedegen voorbereiding van de fase “Uitvoering”. Het fysiek maken van testgevallen voegt dus niets toe aan de zwaarte van de test.

Testsituaties

Definitie

Een testsituatie is een geïsoleerde omstandigheid waaronder het testobject een specifiek gedrag vertoont en die getest moet worden.

Bijvoorbeeld:

“een bestelling van meer dan één boek”

Een andere testsituatie, die daar nauw aan gerelateerd is, zou bijvoorbeeld zijn: “een bestelling van precies één boek”

En als bijvoorbeeld een eventuele korting bij een bestelling pas verleend wordt als het orderbedrag boven een bepaalde drempelwaarde ligt, dan zouden de volgende testsituaties voor de hand liggen: “orderbedrag boven de drempelwaarde” en “orderbedrag beneden de drempelwaarde”

Welke testsituaties precies gedefinieerd worden, hangt af van de **dekking** die nagestreefd wordt en de testontwerptechniek die gekozen wordt om dat te bereiken.

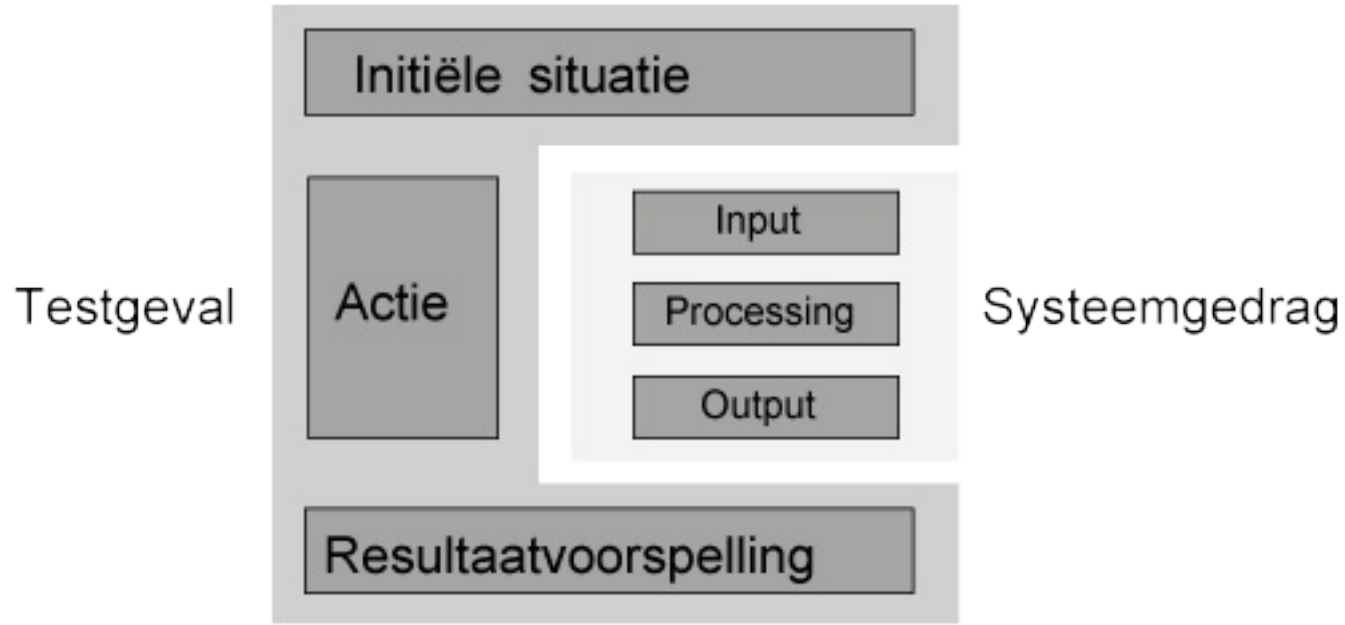
Testgevallen

Definitie

Met een testgeval wordt onderzocht of het systeem onder bepaalde omstandigheden het gewenste gedrag vertoont.

Een testgeval moet dus alle ingrediënten bevatten om dit systeemgedrag te laten gebeuren en om vast te stellen of het goed is of niet. Een bekende manier om het systeemgedrag te beschrijven, is middels “Input Processing Output”. Om het systeemgedrag te kunnen testen, moet een testgeval hier precies op aansluiten en het als het ware “omarmen” (zie figuur 14.3). Vandaar dat in *elk* testgeval, ongeacht de toegepaste testontwerptechniek, de volgende elementen herkenbaar moeten zijn:

- Initiële situatie
  Dit omvat alles wat nodig is om het systeem klaar te zetten om de bedoelde **input** te ontvangen. Hieronder vallen niet alleen de gegevens die voor de ‘processing’ nodig zijn, maar ook de toestand waarin het systeem en zijn omgeving zich in moeten bevinden. Denk bijvoorbeeld aan het instellen van een bepaalde systeemdatum, of het draaien van bepaalde week- en maand-batches die het systeem in een bepaalde toestand brengen.
- Acties
  Dit omvat alle activiteiten die uitgevoerd moeten worden om het systeem tot de **processing** te activeren. Dit kan bestaan uit een simpel commando (“Run …”) of het invoeren van bepaalde gegevens in een scherm. Maar het kan ook een complexe opeenvolging zijn van invoeren van parameters, activeren van een bepaalde functie, manipuleren van andere gegevens, opstarten van een andere functie, enzovoort.
- Resultaatvoorspelling
  Dit omvat alle resultaten die door de tester gecontroleerd moeten worden om vast te stellen of het systeemgedrag conform de verwachting is. Vaak wordt bij resultaatvoorspelling ten onrechte alleen gedacht aan de output die op het scherm verschijnt of die opgeslagen wordt in databases. Maar het systeem kan ook **output** produceren die naar andere systemen of randapparatuur gestuurd wordt. En om te bepalen of het systeem goed werkt, moet er wellicht meer gecontroleerd worden dan alleen uitvoergegevens. Denk bijvoorbeeld aan: “Hoe lang mag het duren tot de output verschijnt?”, “Hoeveel beslag mag er gelegd worden op het geheugen en wordt dit ook weer vrijgegeven?” of “Moet het systeem tussendoor nog signalen of boodschappen produceren, zoals het ‘zandlopertje’ of pieptonen?”



Figuur 14.3: Generieke opbouw van een testgeval, in relatie tot het te testen systeemgedrag.

Anders gezegd, het uitvoeren van een testgeval doorloopt op hoofdlijnen de stappen: “Zet dit klaar → Doe dit → Controleer dat.” In tegenstelling tot een testsituatie – die een geïsoleerd aspect adresseert – is een testgeval een *afgerond geheel* dat als afzonderlijke test uitgevoerd kan worden.

Bij ontwerpen van testgevallen worden eerst *logische testgevallen* gemaakt die vervolgens uitgewerkt worden tot *fysieke testgevallen.* Beide begrippen worden hierna verder toegelicht.

Logische testgevallen

Definitie

Een logisch testgeval beschrijft *in logische termen* de omstandigheden waarin het systeemgedrag onderzocht wordt, door aan te geven *welke testsituaties* door het testgeval gedekt worden.

Neem bijvoorbeeld de eerder genoemde 4 testsituaties over het bestellen van boeken en het eventueel verkrijgen van korting:

- bestelling van meer dan één boek (A1)
- bestelling van precies één boek (A2)
- orderbedrag boven de drempelwaarde (B1)
- orderbedrag beneden de drempelwaarde (B2)

Deze testsituaties worden gedekt door bijvoorbeeld de volgende 2 logische testgevallen:

TG-1: Bestelling van meer dan één boek waarbij het orderbedrag beneden de drempelwaarde blijft.

TG-2: Bestelling van één boek met een bedrag dat boven de drempelwaarde ligt.

Nu is het bij dit eenvoudige en kleine voorbeeld wel erg triviaal. En het kan vast veel handiger en compacter beschreven worden. (Bijvoorbeeld: TG-1 = A1+B2) Maar waar het om gaat, is dat de logische testgevallen op een overzichtelijke manier laten zien:

- wat de testsituaties zijn;
- dat de set testgevallen alle gedefinieerde testsituaties afdekt.

Fysieke testgevallen

Definitie

Een fysiek testgeval is de *concrete uitwerking* van een logisch testgeval, waarbij keuzes gemaakt zijn voor de waarden van alle benodigde invoer en instellingen van de omgevingsfactoren.

Het fysieke testgeval beschrijft in praktische termen wat er gedaan moet worden, waarin de 3 basiselementen van een testgeval herkenbaar zijn: Wat moet er klaargezet worden? (Initiële situatie) Wat moet de tester doen? (Actie) Wat is het verwachte resultaat? (Resultaatvoorspelling).

Bijvoorbeeld kunnen de hierboven beschreven 2 logische testgevallen op de volgende manier fysiek gemaakt worden:

Initiële situatie (geldig voor beide testgevallen):	=	€ 50,00
drempelbedrag	=	€ 50,00
korting	=	10%
prijs boek-01	=	€ 18,50
prijs boek-02	=	€ 25,50
prijs boek-03	=	€ 65,00

TG-1:

Actie:	Bestel boek-01 en boek-02
Resultaatvoorspelling:	Er wordt geen korting verleend. Orderbedrag = € 44,00

TG-2:

Actie:	Bestel boek-03
Resultaatvoorspelling:	Er wordt korting verleend van 10% Orderbedrag = € 58,50 (€ 65,00 - € 6,50)

De stap van logisch testgeval naar fysiek testgeval is meer dan alleen het invullen van concrete waarden. Het is ook een stap van “theoretisch” naar “praktisch”. Bij deze stap moet de tester verder kijken dan de systeemspecificaties waarop de logische testgevallen gebaseerd waren en zich afvragen: “Weet ik nu alles wat ik nodig heb om in de fase “Uitvoering” dit testgeval uit te voeren?” Denk bijvoorbeeld aan het volgende:

- Bij de *initiële situatie*:
  Mag de tester zelf bedenken welke gegevens in de testdatabase aangebracht worden? Of moet er verplicht gebruik gemaakt worden van bestaande testdatabases?
- Bij de *resultaatvoorspelling*:
  Hoe moet vastgesteld worden dat hier aan voldaan wordt? Is het voldoende dat het gegeven correct op het beeldscherm verschijnt, of moet het (ook) via een “Raadpleeg”-functie uit de database opgevraagd worden?

Keuzes bij het uitwerken van het fysiek testontwerp

Het uitwerken van het logisch testontwerp in een fysiek testontwerp is een investering die gedaan wordt in de fase “Specificatie” - buiten het kritieke pad van het project - om een besparing op te leveren in de fase “Uitvoering” - die op het kritieke pad ligt. Het is een keuze van de testorganisatie:

- Hoe ver de tester moet gaan met het fysiek uitwerken van zijn testgevallen.
  Hierbij kunnen twee uitersten onderkend worden:
  – In het ene geval wordt het testgeval niet concreet uitgewerkt, maar blijft het beschreven in abstracte termen. Het wordt aan de tester overgelaten om hieraan tijdens de testuitvoering ‘ter plekke’ een geschikte concrete invulling te geven.
  – In het andere geval wordt tot op het laagste detailniveau voorgeschreven wat de tester precies moet doen: Welke waarde op welke plek in het scherm moet komen en op welke toets vervolgens gedrukt moet worden.
- In welke vorm het resultaat vastgelegd moet worden.
  De meest toegepaste manieren om testgevallen en testscripts vast te leggen, zijn: tekstdocument, spreadsheet, database en in een testwarebeheertool. Het kan ook een mix zijn van deze keuzes. Bijvoorbeeld: het logisch testontwerp (testsituaties en logische testgevallen) is beschreven als tekstdocument, terwijl voor ieder testscript een spreadsheet gemaakt is waarin de fysieke testgevallen uitgewerkt zijn.

Enkele overwegingen bij de keuze hoe ver de testgevallen fysiek uitgewerkt moeten worden:

- Kennis en ervaring van de testers.
  Hierbij gaat het vooral om materiedeskundigheid en kennis over hoe het geautomatiseerde systeem werkt. Naarmate er minder kennis en ervaring is, moet de tester meer ‘voorgekauwd’ krijgen wat hij tijdens testuitvoering precies moet doen.
- Tijdsdruk in de fase “Uitvoering”.
  Hoe minder de testgevallen van tevoren concreet uitgewerkt en voorbereid zijn, hoe meer “denk-“tijd de tester nodig heeft tijdens testuitvoering (en vice versa). Normaal gesproken zit de fase “Uitvoering” op het kritieke pad van het project. Hoe hoger de tijdsdruk in die fase, hoe hoger de noodzaak om daar zo min mogelijk te hoeven doen, dus om zoveel mogelijk goed voorbereid te hebben.
- Overdraagbaarheid naar andere testers.
  Een tester die zelf de testgevallen ontworpen heeft, zal minder uitgebreide informatie nodig hebben om precies te weten hoe hij de test uit moet voeren. Als de test echter uitgevoerd moet kunnen worden door personen die het ontstaan van de testgevallen niet meegemaakt hebben, moeten de fysieke testgevallen en testscripts in principe uitgebreidere informatie bevatten.

Enkele overwegingen om te kiezen in welke vorm het resultaat vastgelegd wordt:

- Beschikbaarheid van tools en kennis over het gebruik ervan.
  Dit is een triviaal argument. Alleen als de testers de beschikking hebben over het daarvoor benodigde tool en weten hoe ze deze moeten gebruiken heeft het zin om voor te schrijven in welke vorm de testgevallen en testscripts vastgelegd moeten worden
- Overdraagbaarheid naar andere omgevingen.

Dit is een belangrijke overweging als er sprake is van verschillende omgevingen waarop de testproducten geïnstalleerd en gebruikt moeten kunnen worden. Tekstdocumenten zijn goed te exporteren naar andere omgevingen. Spreadsheets en databases worden al iets moeilijker. En het overdragen van testproducten die in een testmanagementtool ingebed zijn, vereist al gauw dat de andere omgeving over een identieke installatie van exact dezelfde tool beschikt.

- Koppeling met geautomatiseerde testuitvoering.
  Bij een vergaande testautomatisering worden de tests uitgevoerd door een geautomatiseerde testsuite die gebruik maakt van een specifiek testtool (zie paragraaf 8.5.3 “Soorten testtools”). Dat betekent dat de fysieke testgevallen en testscripts vastgelegd moeten zijn in een vorm die *leesbaar is voor die tool*. Vaak wordt gekozen voor het vastleggen in spreadsheets, databases of in de tool zelf.

Testscripts

Definitie

In een testscript worden meerdere fysieke testgevallen samengebracht om ze efficiënt en eenvoudig uit te kunnen voeren.

Er zijn geen strakke regels die voorschrijven welke testgevallen samengebracht moeten worden in een testscript of hoeveel testscripts er wel of niet zijn toegestaan. De tester heeft daar alle vrijheid in, zolang de testscripts maar gezamenlijk alle testgevallen bevatten. De redenen om bepaalde testgevallen gezamenlijk in één testscript onder te brengen, zijn puur praktisch van aard.
Bijvoorbeeld:

- Ze maken gebruik van dezelfde initiële situatie.
- Bij alle testgevallen moeten dezelfde tijdrovende voorbereidende acties uitgevoerd worden.
- Ze gaan allemaal over dezelfde uitzonderlijke situatie. Bijvoorbeeld: het testscript bevat alle testgevallen over klanten die in het buitenland wonen.
- Er zijn testgevallen met een afhankelijkheid in tijdsvolgorde. Voordat bepaalde testgevallen uitgevoerd kunnen worden, moeten eerst andere testgevallen uitgevoerd zijn.

Vanuit testmanagement-oogpunt zijn testscripts uitstekende ‘te managen’ eenheden. Aan ieder testscript kunnen geplande uren gekoppeld worden. Voortgang kan gemeten en gerapporteerd worden in termen van testscripts.

14.2.2 Dekking, dekkingsvorm en dekkingsgraad

Er zijn veel definities rondom dekking in omloop. De volgende definitie is een eenvoudige die echter uitstekend de essentie weergeeft:

Definitie

Dekking is de verhouding tussen datgene wat getest kan worden en datgene wat met de testset getest wordt.

Dekking zegt dus iets over hoeveel van ‘alle mogelijkheden die getest kunnen worden’ er ook daadwerkelijk getest wordt. Dat wil zeggen dat er twee fenomenen zijn die gezamenlijk bepalen wat de dekking is (zie figuur 14.4):

- Welke mogelijkheden worden er onderscheiden?
  Dit wordt aangeduid met het begrip **dekkingsvorm**. Het geeft aan wat voor soort mogelijkheden beschouwd worden. Deze worden later uitgewerkt in de benodigde *testsituaties*. Zo kan er bijvoorbeeld gekeken worden naar de mogelijke combinaties van paden in een algoritme. Maar in datzelfde algoritme kan ook gekeken worden naar de mogelijkheden binnen ieder beslispunt om het ene of andere pad in te slaan. Dat zijn twee verschillende vormen van dekking (oftewel “dekkingsvormen”).

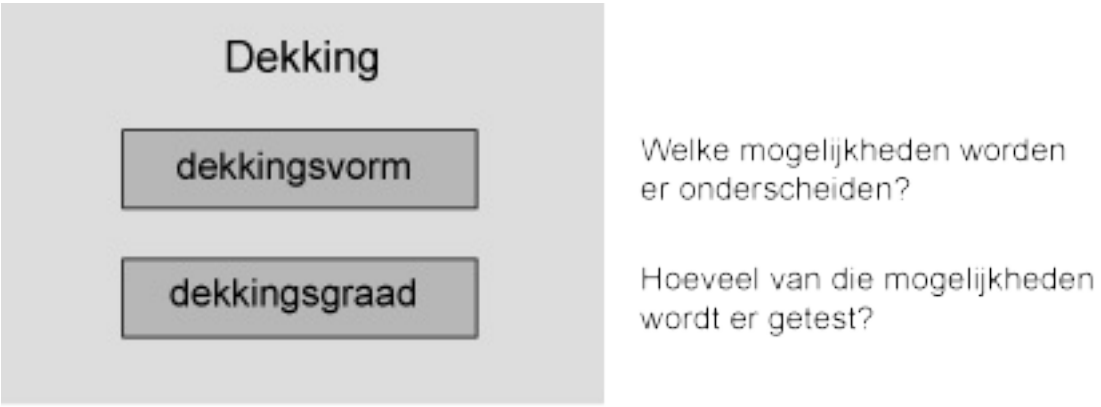
Definitie

Dekkingsvorm is de vorm waarin het afdekken van de te testen situaties die afleidbaar zijn uit de testbasis uitgedrukt wordt.

- Hoeveel (procent) van die mogelijkheden wordt er getest?
  Dit wordt aangeduid met het begrip **dekkingsgraad** en wordt meestal uitgedrukt als percentage. Het geeft aan welk deel van alle mogelijkheden er daadwerkelijk getest is. Het getal (percentage) heeft pas betekenis als het duidelijk is over welke mogelijkheden het hier gaat, dus als het gekoppeld is aan de dekkingsvorm.

Definitie

Dekkingsgraad is het percentage van de door de dekkingsvorm bepaalde testsituaties dat door de test gedekt is.



Figuur 14.4: Onderdelen van het begrip "dekking".

Dit hoofdstuk gaat dieper in op het hoe en waarom van dekking, dekkingsvorm en dekkingsgraad. Eerst wordt kort besproken waarom dekking een zinvol en praktisch begrip is voor testers. Daarna wordt met een uitgebreid voorbeeld toegelicht waarom het noodzakelijk is om verschillende dekkingsvormen te onderscheiden. Tenslotte wordt uitgelegd hoe testontwerptechnieken de verbindende schakel vormen tussen testgevallen en de dekking die daarmee bereikt wordt.

Waarom dekking?

"Dekking" blijkt een zinvol begrip dat testers helpt bij lastige vragen, zoals:

- Als het toch onmogelijk is om alles te testen en we dus gedwongen zijn om slechts een subset van alle mogelijkheden te testen, wat is dan de beste subset?
- Wat is nu eigenlijk precies het verschil tussen iets 'zwaar testen' en iets 'licht testen'? Wat betekent dat concreet voor de testgevallen die we daarvoor nodig hebben?

Dekking heeft alles te maken met de wens om met zo min mogelijk testgevallen zo veel mogelijk fouten te vinden. In plaats van 'zo maar' een subset van mogelijkheden uit te proberen, wordt gestreefd naar een set testgevallen die een *zo groot mogelijke kans* heeft om de aanwezige fouten te vinden.

We kunnen *nooit zekerheid* hebben dat alle fouten gevonden zijn, of dat 60% van de fouten gevonden is. We weten immers niet hoeveel fouten er daadwerkelijk aanwezig zijn. Wat we wel kunnen aantonen, is de dekking die door de test gerealiseerd is. En dat geeft een bepaalde mate van *vertrouwen* dat de kans op resterende fouten in het geteste systeem klein is. Kort gezegd: Hoe hoger de gerealiseerde dekking, hoe kleiner de kans dat er nog onbekende fouten in het systeem zitten.

De keuze om 'zwaarder te testen' vertaalt zich concreet naar de keuze voor een zwaardere dekking. Daarbij zijn in principe drie mogelijkheden:

- Een zwaardere dekkingsvorm;
- Meerdere dekkingsvormen;
- Een hogere dekkingsgraad van een bepaalde dekkingsvorm.

Eén soort dekking of meerdere?

Inmiddels zal duidelijk zijn dat een uitspraak zoals “ik wil testen met een dekkingsgraad van 75%” weinigzeggend is, om de volgende redenen:

- Over wat voor soort dekking gaat het dan?
- Waarom hoeft 25% (en welke) dan niet gedekt te zijn? (Oftewel: Waarom wordt niet gewoon 100% gedekt?)

Deze paragraaf zal hier verder op ingaan en uitleggen

- dat er niet sprake is van *de* dekking, maar van vele vormen van dekking (oftewel “dekkingsvormen”);
- dat de keuze voor zwaarder testen met name ingevuld dient te worden met de keuze voor een andere dekkingsvorm;
- dat het gebruik van een dekkingsvorm impliceert dat een dekkingsgraad van 100% nagestreefd wordt;
- dat in het algemeen de verschillende dekkingsvormen NIET vergeleken kunnen worden in de zin van ‘X is beter dan Y’. Verschillende dekkingsvormen vullen elkaar eerder aan dan dat ze elkaar vervangen.

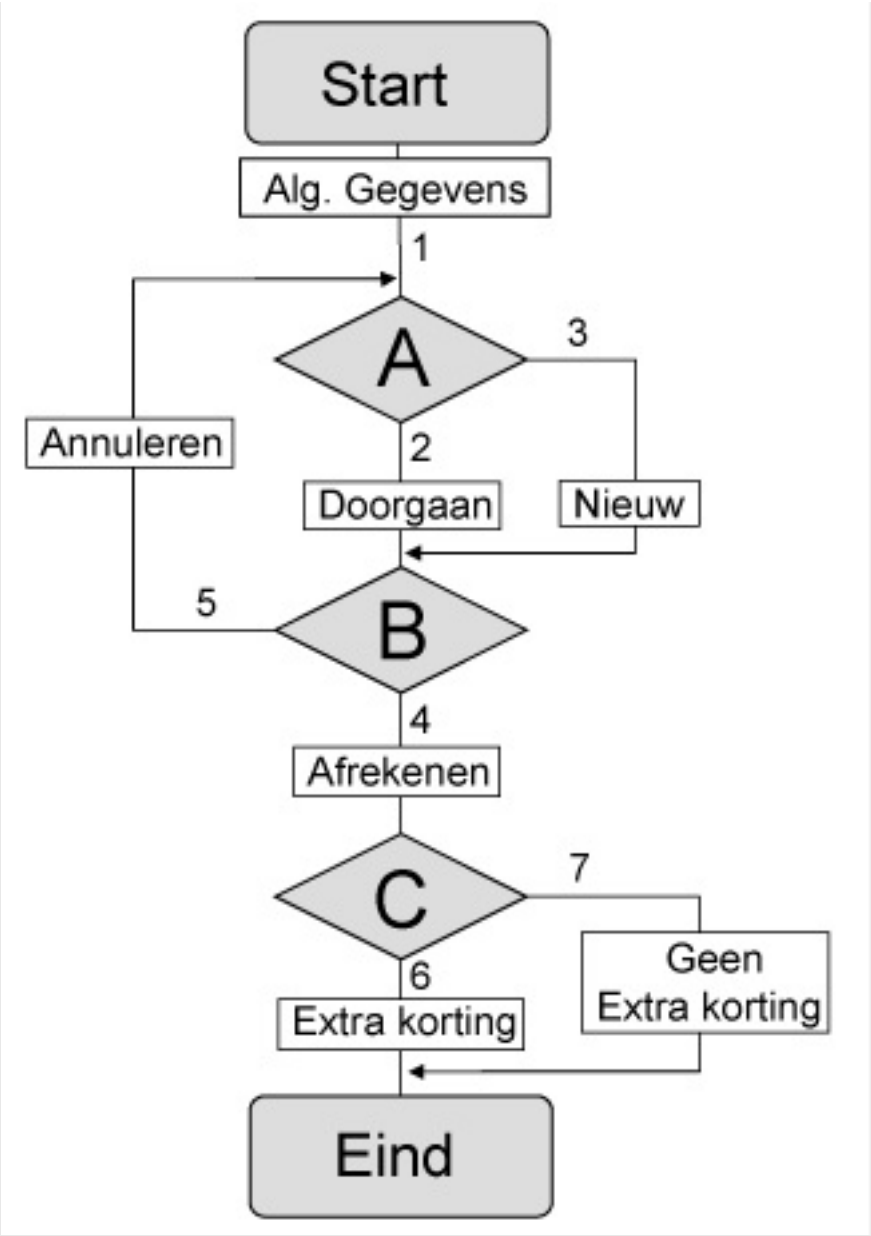
Om dit toe te lichten zal gebruik gemaakt worden van het volgende voorbeeld over een systeem waarmee boeken besteld kunnen worden via internet. Het beschrijft slechts een fractie van wat een dergelijk systeem in werkelijkheid zou kunnen, maar voor ons doel is het voldoende.

Voorbeeld

Onderstaand schema beschrijft het proces voor het bestellen van boeken in ons voorbeeldsysteem. Daarbij is gebruik gemaakt van een intuïtieve schematechniek met beslispunten (A, B, en C) en paden (1 t/m 7). Op de paden van het proces gebeurt in werkelijkheid van alles, maar voor het voorbeeld is dat niet relevant. Het gaat vooral om het feit dat er in het proces *verschillende* mogelijkheden zijn en de vraag welke verschillende mogelijkheden dan getest moeten of kunnen worden.

Het bestelproces zit als volgt in elkaar, waarbij tussen haakjes gerefereerd wordt aan de beslispunten en paden uit het schema:

- Nadat de algemene gegevens van de besteller ingevoerd zijn (1), wordt gekozen (A) of doorgegaan moet worden met een bestaande bestelling (2) of een nieuwe bestelling geïnitieerd moet worden (3). Een bestelling mag meerdere boeken bevatten.
- Nadat de gewenste boeken besteld zijn wordt er beslist (B) of men doorgaat naar de afrekening (4) of dat men de bestelling wil herzien door bestelde boeken te annuleren (5), waarna de bestelling voortgezet kan worden.
- Bij het bepalen van de eindafrekening wordt nog bekeken of de besteller als “goede klant” geldt (C) en recht heeft op extra korting (6) of niet (7).



Figuur 14.5: Schema van het proces "boeken bestellen".

Discussie over testgevallen en dekkingsgraad

Stel dat het hierboven beschreven systeemdeel **grondig** getest moet worden. Wat betekent dat precies? Welke testgevallen zijn daarvoor nodig?

Een testgeval doorloopt het te testen proces van start tot eind, waarbij het diverse beslispunten aandoet en verschillende paden doorloopt. Voor een grondige test zijn we op zoek naar een set testgevallen die zo goed mogelijk slaagt in het bereiken van het doel: "Als er een fout in het proces zit, dan wordt die door deze set testgevallen gevonden."

Bekijk nu de volgende set van twee testgevallen, beschreven door een reeks cijfers die aangeeft welke paden achtereenvolgens doorlopen worden:

TG-1 = 1 2 5 3 4 6

TG-2 = 1 3 4 7

Dankzij de 'loop' in het schema doorloopt testgeval TG-1 al bijna alle paden. Alleen pad 7 wordt nog niet geraakt. Daarvoor is TG-2 toegevoegd.

Hebben we nu met deze twee testgevallen een dekking bereikt van 100%? Als een coverage-tool gebruikt zou worden om de dekking te meten, zou dit inderdaad rapporteren dat 100% dekking behaald is. Toch zouden vele testers opmerken dat deze twee testgevallen nog lang niet alles testen… dat er nog heel wat fouten in kunnen zitten die door deze testgevallen heen glippen. Hebben die testers gelijk? Jazeker!

Kijk bijvoorbeeld naar het volgende:
Stel dat er in pad 5 een fout in het systeem zit. Bij het annuleren van een besteld boek, moet een daaraan gerelateerde korting ongedaan gemaakt worden, maar het systeem *laat deze korting ten onrechte staan*.

Wordt deze fout gevonden met de bovenstaande set van 2 testgevallen? Nee! Het testgeval TG-1 komt weliswaar door pad 5 heen en zal daar dus een te hoge korting krijgen, maar deze verdwijnt vervolgens als het testgeval door pad 3 verder gaat en een nieuwe bestelling van voren af aan begint met korting=0 . Deze fout zou *wel* gevonden worden met een testgeval dat na het doorlopen van pad 5 verder gaat met pad 2. Dan zou doorgegaan worden met dezelfde bestelling en bij afrekening wel een te hoge korting zichtbaar zijn.

Er is hier dus sprake van een type fout dat niet zomaar 'in een pad optreedt', maar pas optreedt bij een speciale combinatie van twee opeenvolgende paden. De twee testgevallen TG-1 en TG-2 hebben dus wel een dekkingsgraad van 100% in de zin van "alle paden gedekt", maar hebben daarmee nog niet "alle combinaties van twee opeenvolgende paden" gedekt. Bij een set testgevallen die dat wel dekt, spreekt men van het behalen van "testmaat-2".

Testmaat-2 = de zekerheid dat alle combinaties van 2 opeenvolgende paden gedekt zijn.

Voor '100% van testmaat-2' moeten ook de padcombinaties 2-4 en 3-5 nog gedekt worden. Dat kan bereikt worden door het toevoegen van het volgende testgeval:

TG-3 = 1 3 5 2 4 7

(Opgemerkt kan worden dat hiermee testgeval TG-2 niet meer nodig is om testmaat2 te bereiken.)

Het is overigens een heel gebruikelijk fenomeen dat verschillende paden elkaar beïnvloeden zodat fouten pas optreden bij een combinatie van dergelijke paden. Zo'n beinvloeding kan functioneel van aard zijn (zoals het bovenstaande voorbeeld van een te hoge korting), maar kan ook technisch van aard zijn. Een voorbeeld daarvan is:

- Bij het terugboeken van een eerder gedane korting in pad 5, treedt een buffer-overflow op waardoor de geheugenlocatie van de variabele "extra korting" overschreven wordt. Als een testgeval dat door pad 5 loopt daarna door pad 7 gaat (waar geen extra korting verleend wordt), zal er van deze fout niets merkbaar zijn. Als het testgeval echter verder zou gaan met pad 6, zou dit leiden tot het toekennen van een bizarre extra korting of mogelijk zelfs tot een crash van de applicatie.

Zo kunnen er natuurlijk ook fouten bestaan die pas optreden bij een specifieke combinatie van drie opeenvolgende paden. Dit type fouten wordt pas *met zekerheid* gevonden als de set testgevallen testmaat-3 bereikt. Enzovoorts… Hoe hoger de testmaat die bereikt wordt, hoe groter de zekerheid dat de meer *uitzonderlijke* fouten gevonden worden.

Het begrip "testmaat" en zijn toepassing wordt in meer detail toegelicht in de paragraaf over "Dekken van paden".

Is dit het dan: Hoe hoger de testmaat, hoe grondiger de test? De maximale testmaat is 'oneindig', maar die kan natuurlijk praktisch gezien niet bereikt worden. Maar zou je (theoretisch) mogen concluderen dat je met testmaat 'oneindig' alle fouten gaat vinden? Nee! Zelfs dat niet! Onderstaand voorbeeld laat een fout zien die zelfs met een testset van testmaat 'oneindig' niet met zekerheid gevonden wordt. En de beschreven fout is niet eens vergezocht:

Een "goede klant" krijgt extra korting. Dit wordt vastgesteld in beslispunt C, waarin gekeken wordt of de klant veel boeken heeft besteld of voor een hoog bedrag. Exacter gespecificeerd:

ALS (aantal boeken > 8 OF bedrag ≥ €250) DAN extra korting

Dan zijn er dus 4 verschillende situaties denkbaar waarin al of niet extra korting verleend wordt. Deze staan hieronder weergegeven:

aantal boeken	>	8 ; bedrag	≥	250 →	WEL extra korting	(1)
	>		<		WEL	(2)
	≤		≥		WEL	(3)
	≤		<		NIET	(4)

Stel dat er een fout in het systeem zit, waardoor je bij situatie (3) geen extra korting krijgt. Dus als je weinig maar wel dure boeken bestelt, krijg je ten onrechte geen korting.

De hierboven beschreven fout (situatie 3) treedt op in pad 6 waar je WEL extra korting hoort te krijgen. Er zijn echter nog twee andere situaties die ook bij pad 6 horen. Onze testset van testmaat 'oneindig' is wel heel vaak door pad 6 gekomen, maar dat garandeert nog niet dat minstens één testgeval situatie 3 uitgeprobeerd heeft. Het zou *bij toeval* best zo kunnen zijn, maar we hebben *niet de zekerheid* dat situatie 3 geraakt wordt. Waarom niet? Omdat de testset van testmaat 'oneindig' daar *niet voor ontworpen is*. De testset was ontworpen om combinaties van paden af te dekken, maar daar heeft deze fout niets mee te maken. De fout zou wel met zekerheid gevonden zijn, als een testset gebruikt werd met een *ander soort dekking*, namelijk een dekking die zich richt op hoe een complex beslispunt de verschillende mogelijkheden afhandelt. Bijvoorbeeld door alle mogelijke combinaties van alle condities af te dekken. In bovenstaand voorbeeld komt dat neer op alle 4 de genoemde situaties. Een dergelijke *dekkingsvorm* heet "multiple condition coverage". Een andere dekkingsvorm is bijvoorbeeld de "decision coverage", waarmee slechts de twee mogelijke uitkomsten van het beslispunt (WEL of GEEN extra korting) afgedekt worden. Dit is een minder zware dekking dan multiple condition coverage, waarmee de bovenstaande fout *niet* met zekerheid gevonden zou worden.

Wat is nu het beste? Kiezen we voor het afdekken van paden of voor het afdekken van beslispunten? Natuurlijk is het doen van allebei het grondigst. Maar stel dat dat niet toegestaan is (bijvoorbeeld omdat dat te kostbaar is). Stel dus dat we **moeten kiezen**, wat is dan het beste? Concreter geformuleerd:

Welke dekking is beter: "testmaat-2" of "multiple condition coverage"?

De vraag lijkt simpel, maar dat is het niet. De ene dekking zou pas zinvol "beter" genoemd kunnen worden, als hij minstens alle fouten vindt die de andere dekking vindt plus nog wat extra fouten. Maar bovenstaand voorbeeld heeft laten zien dat met "multiple condition coverage" een bepaalde fout **wel** gevonden wordt, die met testmaat-2 **niet** gevonden wordt (namelijk de extra korting voor weinig maar dure boeken). Maar ook omgekeerd (namelijk de korting die ten onrechte blijft staan bij annulering). Het beste antwoord is dus:

Er is geen sprake van "beter"!

De verschillende dekkingsvormen vinden *andere* fouten.

Tip

Als er meerdere dekkingsvormen in aanmerking komen waaruit gekozen moet worden, overleg dan met de opdrachtgever. Analyseer op welk gebied de grootste risico's liggen en kies dan welke dekkingsvorm daar het best bij past. Zorg er vervolgens voor dat de opdrachtgever zich bewust is van het feit dat er risico's gelopen worden op het gebied dat te maken heeft met de niet-gekozen dekkingsvorm.

Bijvoorbeeld bij de eerder genoemde vraag of beter "testmaat-2" of "multiple condition coverage" gekozen kan worden, moeten de tester en opdrachtgever zich typisch het volgende afvragen:

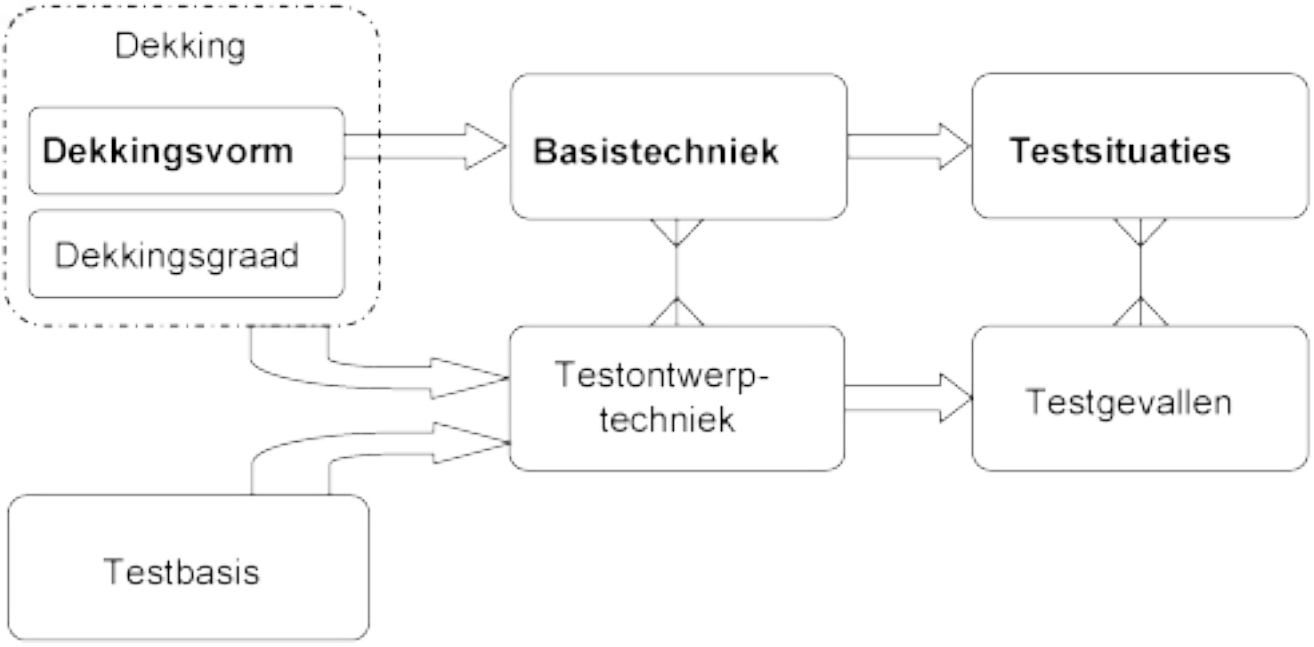
Is het een proces waarbij veel verschillende soorten acties uitgevoerd worden waarvan de gevolgen elkaar mogelijk beïnvloeden? Dan is de paden-dekking testmaat-2 een goede keuze. Maar is het bijvoorbeeld een proces waarbij steeds na ieder beslispunt een bepaalde korting wel of niet toegekend wordt, dan zijn combinaties van paden niet zo interessant, omdat de kortingen elkaar niet beïnvloeden. Daarentegen zou de opdrachtgever juist de mogelijke situaties waaronder de korting wel of niet toegekend wordt, van groot belang kunnen achten. In dat geval is de dekkingsvorm "multiple condition coverage" een verstandige keuze.

14.2.3 Testontwerptechniek en basistechniek

Technieken als verbinding tussen dekking en testgevallen

Zoals in paragraaf 14.1 is beschreven, wordt een testontwerptechniek gebruikt om vanuit een bepaalde testbasis de benodigde testgevallen af te leiden die de beoogde dekking behalen. Dit afleiden van de testgevallen verloopt niet

rechtstreeks, maar loopt via het bepalen van de gewenste **dekkingsvorm**(en) en het afleiden van de **testsituaties** waarmee die dekkingsvorm gehaald wordt (zie figuur 14.6). Dit wordt hieronder verder toegelicht.



Figuur 14.6: Afleiden van testsituaties vanuit dekkingsvorm.

De gewenste dekking wordt concreet uitgedrukt in de geselecteerde dekkingsvormen. Iedere **dekkingsvorm** vereist een bepaald soort informatie in de testbasis, bijvoorbeeld een gestructureerd stroomdiagram met paden en beslispunten. Voor een bepaalde dekkingsvorm kan een standaard werkwijze opgesteld worden om de benodigde testsituaties af te leiden. Dit wordt hier een **basistechniek** genoemd.

Definitie

Basistechniek is de wijze van afleiden van de testsituaties uit de testbasis die tot de gewenste dekkingsvorm leidt.

Overigens zou voor sommige basistechnieken het afleiden van de benodigde testsituaties eventueel door een tool uitgevoerd kunnen worden.

Paragraaf 14.3 zal dieper ingaan op de belangrijkste dekkingsvormen en de beschikbare basistechnieken.

Hoe zijn technieken gerelateerd aan kwaliteitsattributen/ testvormen?

Het begrip **testontwerptechniek** is nauw gerelateerd aan de **testvorm** die uitgevoerd wordt en daarmee aan het **kwaliteitsattribuut** dat met die testvorm getest wordt. Een testontwerptechniek beschrijft de testbasis die nodig is, de dekkingsvormen die nagestreefd worden en de basistechnieken die toegepast worden om deze dekkingsvormen te behalen.

Paragraaf 14.4 zal de meest voorkomende en praktisch bruikbare testontwerptechnieken beschrijven en met een voorbeeld uitwerken.

Een **basistechniek** is in principe NIET gerelateerd aan een kwaliteitsattribuut.

Bijvoorbeeld het dekken van paden met testmaat-2 (een dekkingsvorm) kan toegepast worden op het testen van functionaliteit, maar bijvoorbeeld ook op beveiliging, mits de beveiligingsspecificaties in termen van paden en beslispunten beschreven zijn.

Ter verduidelijking van het onderscheid en de samenhang tussen testontwerptechniek, basistechniek en kwaliteitsattribuut, worden als voorbeeld onderstaande twee testontwerptechnieken beschreven:

- De Procescyclustest (zie paragraaf 14.4.8) is een testontwerptechniek voor het testen van het kwaliteitsattribuut “inpasbaarheid”. De benodigde testbasis is de beschrijving van de AO (Administratieve Organisatie) in termen van

paden en beslispunten. De beoogde dekkingsvorm is “pad-dekking testmaat-2”.
- De Algoritmetest (zie www.tmap.net) is een testontwerptechniek voor het testen van het kwaliteitsattribuut “functionaliteit”. De testbasis bestaat uit de algoritme-beschrijving van een stuk programmacode in termen van paden en beslispunten. De beoogde dekkingsvorm is “pad-dekking testmaat 2”.

De twee beschreven testontwerptechnieken hebben elk een ander doel (ander kwaliteitsattribuut dat getest wordt) en werken op heel verschillende testobjecten (AO-procedures en programma-code). Maar beide testontwerptechnieken streven dezelfde dekkingsvorm na (pad-dekking testmaat-2) en passen daarom dezelfde basistechniek toe. In feite hoeft een tester dus slechts één basistechniek te leren (pad-dekking testmaat-2) om beide testontwerptechnieken te kunnen toepassen.

14.3 Dekkingsvormen en basistechnieken

14.3.1 Inleiding

Deze paragraaf geeft een overzicht van de belangrijkste dekkingsvormen. Tabel 14.1 geeft bij iedere dekkingsvorm uit dit hoofdstuk een korte omschrijving. Indien van toepassing wordt met een trefwoord aangegeven hoe binnen de dekkingsvorm de zwaarte van de dekking gevarieerd kan worden. Een andere manier om de gewenste dekking te verzwaren, is het combineren van verschillende dekkingsvormen. Dit wordt aan het eind van deze paragraaf verder toegelicht.

Dekkingsvorm	Omschrijving	Variatie in zwaarte
Paden	Dekken van de variaties in het procesverloop in termen van combinaties van paden. Als testbasis is een schema nodig van beslispunten en paden.	testmaat-N
Beslispunten	Dekken van de verschillende mogelijkheden binnen een beslispunt om tot de uitkomsten WAAR en ONWAAR te komen. Als testbasis is een formele beschrijving van het beslispunt nodig, waarbij de condities binnen het beslispunt verbonden zijn met EN, OF en NIET.	condition/decision modified condition/ decision multiple condition
Equivalentieklassen	Het waardebereik van een parameter wordt opgedeeld in klassen waarbij een ander systeemgedrag optreedt. Het systeem wordt getest met minimaal één waarde uit iedere klasse.	—
Pairwise testing	Combineren van een bestaande set testsituaties, die bijvoorbeeld uit “equivalentieklassen” of “beslispunten” ontstaan zijn. Pairwise testing combineert alle mogelijkheden van iedere set van twee parameters.	N-wise testing
Orthogonale arrays	Zie “pairwise testing”	strength N
Grenswaarde analyse	Een grenswaarde bepaalt de overgang van de ene equivalentieklasse naar de andere. Grenswaarde analyse test de grenswaarde zelf plus de waarde direct hierboven en direct hieronder.	grenswaarde plus één of plus twee waarden rondom de grens
CRUD	Dekken van alle basisbewerkingen (Create, Read, Update, Delete) op alle entiteiten.	—
Operational profiles	Simuleren van het realistisch gebruik van het systeem, door een *opeenvolging van transacties* uit te voeren, die statistisch verantwoord samengesteld is.	—
Load profiles	Simuleren van een realistische belasting van het systeem in termen van *hoeveelheden* gebruikers en/of transacties.	—
Goedpaden/Foutpaden	Dekken van het goedpad en het foutpad bij iedere gedefinieerde foutsituatie.	Alleen goedpaden of alleen foutpaden
Afvinklijst	Afvinken van een lijst zonder structuur. Ieder element in de lijst wordt rechtstreeks getest door minimaal één testgeval.	—

Tabel 14.1: Korte omschrijving van de belangrijkste dekkingsvormen.

Deze dekkingsvormen worden in meer detail toegelicht in de volgende paragrafen. Waar nodig wordt de basistechniek uitgelegd om de benodigde testsituaties voor de betreffende dekkingsvorm af te leiden.

Het is niet bedoeld om een uitputtende lijst te produceren van dekkingsvormen en bijbehorende basistechnieken. Zo bestaat er bijvoorbeeld ook het dekken van “state transition diagrams” en het toepassen van “evolutionaire algoritmen”. Deze zijn echter vooral van toepassing op zogenaamde embedded systemen en worden onder andere beschreven in het boek “Testing Embedded Software” [Broekman, 2003].

Verzwaring van de dekking door dekkingsvormen te combineren

De gewenste dekking van een test kan verzwaard worden door meerdere dekkingsvormen in de test te combineren.

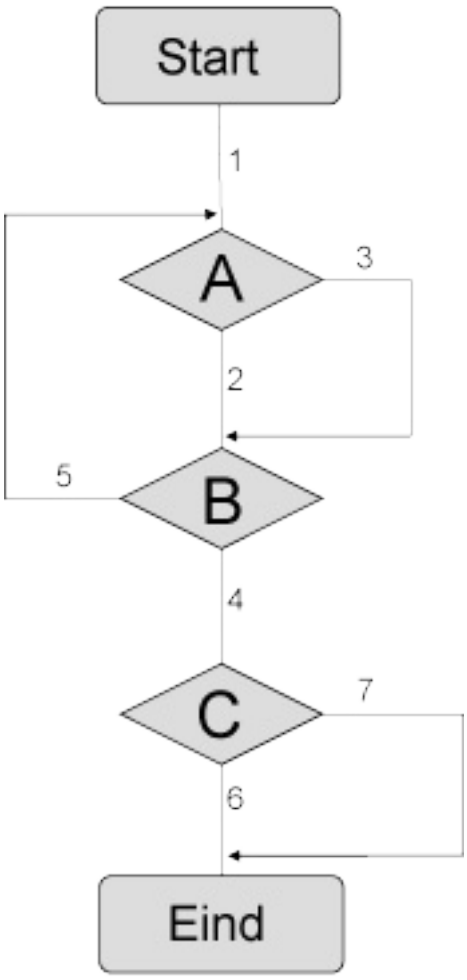
Grenswaarde analyse en **pairwise testing** zijn typisch dekkingsvormen die gebruikt worden om extra verzwaring te geven aan een toegepaste dekkingsvorm. Ze zijn met name geschikt om te combineren met het dekken van beslispunten en van equivalentieklassen.

Ook andere dekkingsvormen kunnen onderling gecombineerd worden, mits de benodigde informatie als testbasis beschikbaar is. Iedere dekkingsvorm afzonderlijk levert een aantal testsituaties, die gezamenlijk afgedekt worden in de testgevallen. Enkele voorbeelden van bruikbare combinaties van dekkingsvormen zijn:

- Paden + Equivalentieklassen
  Bijvoorbeeld: Het procesverloop van “Orders – Leveren - Betalen” is gespecificeerd in een proces-schema. Dit wordt getest met *paden-dekking testmaat2*. Variaties in soorten orders en in manieren van betalen worden ontworpen met behulp van *equivalentieklassen*.
  Dit kan nog weer extra verzwaard worden door grenswaardenanalyse en/of pairwise testing toe te passen.
- CRUD + Beslispunten
  Bijvoorbeeld: Van de gegevens “order”, “levering” en “betaling” wordt een CRUD matrix samengesteld. Hiermee wordt de levensloop van deze gegevens getest. Er zijn integriteitsregels gespecificeerd voor de mogelijke bewerkingen van de gegevens. Deze zijn formeel beschreven, zoals bijvoorbeeld “
  ALS conditie-A OF (conditie-B EN conditie-C) DAN bewerking niet toegestaan”.
  Deze integriteitsregels worden afgedekt door het toepassen van modified condition/decision coverage.

14.3.2 Paden

Het dekken van paden is van toepassing als het gedrag van het systeem beschreven is met behulp van beslispunten en paden. Figuur 14.7 laat hiervan een voorbeeld zien, dat ook gebruikt is als voorbeeld in paragraaf 14.2.2.



Figuur 14.7: Voorbeeld van een specificatie van het systeemgedrag in termen van beslispunten en paden.

Een dergelijk schema van beslispunten en paden laat op een gestructureerde manier zien hoe het proces loopt van start tot eind en wat de verschillende mogelijkheden in dat procesverloop zijn: Bij elk beslispunt kan het proces verschillende kanten op gaan, aangegeven door de verschillende paden die vanuit dat beslispunt verder gaan. De voorwaarden waaronder het ene of het andere pad ingeslagen wordt, zijn beschreven in de beslispunten zelf.’

Het doel van de hier beschreven dekkingsvorm is om de *variaties in het procesverloop* af te dekken die volgens het schema mogelijk zijn. De **testsituaties** worden in dit geval beschreven door aan te geven welke paden uit het schema achtereenvolgens doorlopen moeten worden.

Let wel, dergelijke schema's met beslispunten en paden hoeven niet alleen maar te gaan over de *functionaliteit* van het systeem. Ook beveiligingsprocessen of werkprocedures in bedrijfsprocessen kunnen met dergelijke schema's beschreven zijn. Daarmee is de hier beschreven basistechniek toepasbaar voor meerdere testvormen.

Ook het abstractieniveau doet niet ter zake: Het is toepasbaar op zowel gedetailleerd niveau (code algoritme) als op overkoepelend systeemniveau, zolang de informatie over het gewenste systeemgedrag maar in de gestructureerde vorm van beslispunten en paden gegeven is.

Lichtere of zwaardere dekking: de testmaat

Bij het dekken van paden zijn verschillende zwaarteniveaus mogelijk. Hoe hoger het zwaarte-niveau, hoe groter de foutvindkans. Dit wordt hieronder toegelicht.

De lichtste vorm van dekking van paden, is slechts de garantie dat elk pad een keer doorlopen is. De **testsituaties** bestaan in dit geval uit elk afzonderlijk pad. In ons voorbeeld (figuur 14.7) zijn dat de testsituaties:

1; 2; 3; 4; 5; 6; 7

Een **testgeval** is een afgerond geheel dat het procesverloop van "Start" tot "Eind" doorloopt. Bovengenoemde testsituaties kunnen bijvoorbeeld met de volgende twee logische testgevallen afgedekt worden:
TG-1 = 1 2 5 3 4 6
TG-2 = 1 3 4 7

Hiermee worden dus alle fouten gevonden die zonder meer in een bepaald pad optreden. Echter, fouten die pas optreden bij een specifieke *combinatie van processtappen*, worden hiermee niet met zekerheid gevonden. Zo kan in ons voorbeeld een bepaalde fout aanwezig zijn, die pas optreedt als pad-2 direct na pad-5 uitgevoerd wordt (zie ook het uitgewerkte voorbeeld in 14.2.2). Dat komt overeen met een testsituatie "5-2" en die is *niet* door de bovenstaande twee testgevallen gedekt. Een dergelijk soort fout wordt pas gevonden als de variaties in het procesverloop op een *zwaardere* manier afgedekt worden. Normaal gesproken betekent dat, dat er meer of complexere testgevallen nodig zijn. En stel dat er een fout bestaat die pas optreedt als "pad-2 direct na pad-5 uitgevoerd wordt, dat op zijn beurt direct na pad-3 uitgevoerd is". Dat komt overeen met een testsituatie "3-5-2". Om dit soort fouten te vinden, moet nog zwaarder getest worden.

De mate van lichter of zwaarder dekken wordt weergegeven met het begrip testmaat:

Definitie

Testmaat-N is de zekerheid dat alle combinaties van N achtereenvolgende paden afgedekt zijn.

De testmaat loopt in principe van 1 tot oneindig. Hoe hoger de testmaat, hoe groter de zekerheid dat ook fouten die optreden bij een complex samenstel van processtappen worden gevonden. Een hogere testmaat impliceert een lagere testmaat. In andere woorden: Een hogere testmaat vindt in ieder geval alle fouten die met een lagere testmaat gevonden worden plus mogelijk nog andere fouten.

Het afleiden van de testsituaties voor testmaat-1 is triviaal. De basistechniek voor het behalen van testmaat-2 wordt direct hieronder beschreven. Daarna wordt toegelicht hoe hogere testmaten eenvoudig afgeleid kunnen worden vanuit testmaat-2.

Techniek voor het behalen van testmaat-2

Ongeacht de testmaat, is als uitgangspunt voor deze techniek een **testbasis** nodig die het systeemgedrag beschrijft in termen van beslispunten en paden.

Vervolgens worden de volgende stappen uitgevoerd:

1. Beslispunten & Paden.
   In het processchema de beslispunten benoemen (A, B, enz.) en de paden nummeren, zoals in figuur 14.7.
   Per beslispunt opsommen van
   a. Inkomende paden ("IN")

   b. Uitgaande paden (“UIT”)
2. Padcombinaties.
   Uitwerken van alle combinaties van “IN” en “UIT” bij ieder beslispunt.
   Bij een aantal van P inkomende paden en Q uitgaande paden, leidt dit tot P maal Q padcombinaties.
3. Logische testgevallen.
   Samenstellen van logische testgevallen, totdat alle padcombinaties afgedekt zijn. Ieder logisch testgeval begint bij “Start” en eindigt bij “Eind” en doorloopt daarbij diverse padcombinaties.

De eerste twee stappen leiden tot de **testsituaties** in het testontwerp. Het zijn alle variaties (in dit geval: padcombinaties) die getest moeten worden om testmaat-2 te behalen.

Stap 3 leidt tot een set **logische testgevallen** die gegarandeerd alle testsituaties afdekt. Het is de vrijheid van de tester om een set testgevallen samen te stellen die hieraan voldoet. Eventueel kan met behulp van een *cross-reference matrix* aangetoond worden dat met de gekozen set testgevallen alle testsituaties gedekt worden. In principe zijn er twee manieren om tot een dekkende set testgevallen te komen:

- Werk vanuit het processchema. Definieer een testgeval door op een bepaalde manier door het proces te lopen van “Start” tot “Eind”. De tester is in principe vrij om te kiezen op welke manier het proces doorlopen wordt. Streep uit de lijst padcombinaties iedere combinatie weg die in dit testgeval voorkomt. Herhaal dit proces totdat de lijst padcombinaties volledig weggestreept is.
- Werk vanuit de lijst padcombinaties. Begin met een padcombinatie die start vanuit “Start” (bijvoorbeeld “1-3”). Zoek een volgende padcombinatie die begint met het pad waar de vorige mee eindigde (dus bijvoorbeeld een “3-5”). In feite net als het leggen van dominostenen. Ga door met het zoeken van een volgende padcombinatie totdat “Eind” bereikt is. Uiteraard wordt zoveel mogelijk een nog niet eerder gedekte padcombinatie gebruikt.

De stappen worden nu toegelicht met het voorbeeld van figuur 14.7.

Uitgediept

Uitwerking van testmaat-2 voor het schema van figuur 14.7

1. Beslispunten & Paden
   A: IN: 1,5
      UIT: 2,3
   B: IN: 2,3
      UIT: 4,5
   C: IN: 4
      UIT: 6,7
2. Padcombinaties
   A: 1-2; 1-3; 5-2; 5-3
   B: 2-4; 2-5; 3-4; 3-5
   C: 4-6; 4-7
3. Logische testgevallen
   TG-1 = 1-2-5-3-4-6 (Streep zelf de hiermee gedekte padcombinaties weg.)
   TG-2 = 1-3-5-2-4-7 (Volgt eenvoudig uit de overgebleven padcombinaties.)

Cross-reference matrix (optioneel)

	TG-1	TG-2
1-2	X	
1-3		X
5-2		X
5-3	X	
2-4		X
2-5	X	
3-4	X	
3-5		X
4-6	X	
4-7		X

Afleiden van testsituaties voor hogere testmaten

Voor hogere testmaten wordt het eenvoudige mechanisme gehanteerd:

- Ga uit van de lijst padcombinaties van de voorgaande testmaat.
- Breidt elke padcombinatie uit met iedere mogelijke volgende stap in het procesverloop. Dus stel dat na pad P, er drie mogelijke vervolgstappen zijn, zeg Q, R en S. Dan wordt iedere padcombinatie die eindigt op P uitgebreid tot drie nieuwe 'hogere' padcombinaties: ..PQ en ..PR en ..PS.

Het zou geformuleerd kunnen worden als:

Testmaat-(N+1) = Testmaat-N + "1 stap verder in het procesverloop"
In onderstaand voorbeeld is dit concreet uitgewerkt.

Uitgediept

Testmaat-3 voor het schema van figuur 14.7
Uit het schema is eenvoudig het volgende af te leiden:
Pad 1 is het startpunt voor ieder testgeval;
Paden 2 en 3 worden gevolgd door paden 4 en 5;
Pad 5 wordt gevolgd door paden 2 en 3;
Pad 4 wordt gevolgd door paden 6 en 7;
Paden 6 en 7 zijn eindpaden en hebben geen vervolg.
Hiermee kunnen de padcombinaties van testmaat-2 rechtstreeks uitgebreid worden tot testmaat-3:

Padcombinaties van testmaat-2		Uitgebreid tot testmaat-3
A:	1-2	1-2-4; 1-2-5
	1-3	1-3-4; 1-3-5
	5-2	5-2-4; 5-2-5
	5-3	5-3-4; 5-3-5
B:	2-4	2-4-6; 2-4-7
	2-5	2-5-2; 2-5-3
	3-4	3-4-6; 3-4-7
	3-5	3-5-2; 3-5-3
C:	4-6	Geen uitbreiding en reeds afgedekt.
	4-7	Geen uitbreiding en reeds afgedekt.

Op dezelfde wijze kunnen de testsituaties van testmaat-3 uitgebreid worden tot testmaat-4.

14.3.3 Beslispunten

Inleiding; basisprincipes van Booleaanse algebra

Bij vrijwel ieder systeem is sprake van *beslispunten*, waar het systeemgedrag *verschillende kanten* op kan, afhankelijk van de uitkomst van zo'n beslispunt.

Definitie

Een beslispunt is een samenstelling van één of meer condities die de voorwaarden definieert voor de verschillende

mogelijkheden in het daaropvolgende systeemgedrag.

De verschillende condities bepalen gezamenlijk de uitkomst (Engels: “decision”) van het beslispunt. De wijze waarop een conditie bijdraagt aan de uitkomst wordt weergegeven met termen als “EN” of “OF”. Er is een speciaal soort wiskunde – de zogenaamde Booleaanse Algebra, of propositielogica – voor het manipuleren van dit soort constructies. Dit hoofdstuk maakt gebruik van de theorie van Booleaanse algebra, maar is niet bedoeld als leerboek hiervan. De geïnteresseerde lezer wordt verwezen naar de talloze literatuur over dit onderwerp. Hieronder volgen de belangrijkste basisprincipes van Booleaanse algebra die nodig zijn voor de technieken om beslispunten te dekken.

Neem bijvoorbeeld het volgende beslispunt dat uit slechts één conditie bestaat:

ALS (aantal boeken > 8) DAN extra korting

Beslispunten die bestaan uit dergelijke **enkelvoudige condities** leiden tot twee testsituaties, namelijk de situatie waarin de conditie waar is en de situatie waarin de conditie onwaar is. In de Booleaanse algebra wordt 0 gebruikt om aan te geven dat iets onwaar is; een 1 wordt gebruikt als iets waar is. In ons voorbeeld zijn dit de volgende testsituaties:

Testsituatie	**1**	**2**
aantal boeken	> 8	≤ 8
Resultaat	**waar (1)**	**onwaar (0)**

Beslispunten kunnen ook bestaan uit combinaties van condities, de zogenaamde **samengestelde conditie**. Vergelijk de volgende samengestelde condities:

ALS (aantal boeken > 8 OF bedrag ≥ €250) DAN extra korting en
ALS (aantal boeken > 8 EN bedrag ≥ €250) DAN extra korting

Vaak wordt een verkorte schrijfwijze gehanteerd door de condities te vervangen door een hoofdletter (A, B, …). De twee hierboven genoemde beslispunten worden dan afgekort tot

A OF B en
A EN B.

Ook een samengestelde conditie is waar of onwaar. Dit is echter afhankelijk van het feit of de afzonderlijke condities waar of onwaar zijn en de wijze waarop de condities verbonden zijn (de zogenaamde **operatoren**): door een EN dan wel een OF. Uitgaande van twee condities levert dit de volgende mogelijke combinaties op:

A	B
1	1
1	0
0	1
0	0

Dit wordt wel de **volledige beslistabel** genoemd.

In de 0-0 situatie zijn beide beweringen onwaar. In de 0-1 situatie en de 1-0 situatie is slechts één van beide beweringen waar en in de 1-1 situatie zijn beide waar. Het uiteindelijke resultaat in elk van de 4 situaties hangt af van de operator “EN” of “OF”: Bij een “EN” is het eindresultaat van twee condities alleen waar indien beide afzonderlijke condities waar zijn; in alle andere gevallen is het eindresultaat onwaar. Bij een “OF” geldt het omgekeerde: Het eindresultaat is alleen onwaar indien beide afzonderlijke condities onwaar zijn; in alle andere gevallen is het eindresultaat waar.

Een veelgebruikte manier om de uitkomsten te laten zien van alle situaties van een volledige beslistabel, is de **waarheidstabel**. Hieronder zijn de waarheidstabellen aangegeven voor de samengestelde condities “A OF B” en “A EN B”.

A	B	A OF B
1	1	1
1	0	1
0	1	1
0	0	0

A	B	A EN B
1	1	1
1	0	0
0	1	0
0	0	0

Voor een beslispunt dat niet uit twee maar uit drie condities bestaat, is de volledige beslistabel:

A	B	C
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Het zal duidelijk zijn dat het aantal situaties exponentieel toe neemt met het aantal condities. Bij zes condities zijn er al 64 situaties.

Als er sprake is van verschillende operatoren (EN en OF) in een samengestelde conditie, hangt de uitkomst af van welke operator het eerst uitgevoerd wordt. Om geen verwarring hierover te laten bestaan, kan dit het best met behulp van haakjes aangegeven worden. Bij afwezigheid van haakjes geldt de regel dat de EN voor de OF gaat. Als voorbeeld is hieronder de waarheidstabel weergegeven voor de samengestelde conditie “(A OF B) EN C”.

A	B	C	(A OF B) EN C
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Hoewel er vele verschillende Booleaanse operatoren bestaan, blijkt iedere operator teruggebracht te kunnen worden tot een combinatie van drie elementaire operatoren: EN; OF; NIET. Dat wil zeggen, dat als de tester weet hoe hij met deze drie operatoren om moet gaan, hij *ieder beslispunt* aan kan. (De operator “NIET” wordt hier niet verder behandeld. Het effect van deze operator is triviaal: De uitkomst “1” wordt omgezet in “0” en omgekeerd.)

De bekendste dekkingsvormen m.b.t. beslispunten

De volledige beslistabel definieert alle mogelijke combinaties van de afzonderlijke condities. Het testen van al deze mogelijkheden is dus de zwaarste dekkingsvorm bij een beslispunt en wordt ook wel “multiple condition coverage” genoemd. Daarnaast zijn er verschillende dekkingsvormen waarmee een subset van de volledige beslistabel getest wordt. Het gaat dan niet om een willekeurige subset, maar om een subset die een specifiek doel bereikt. Tabel 14.2 beschrijft in het kort de bekendste dekkingsvormen en hun zwaarte ten opzichte van elkaar. Een zwaardere dekkingsvorm ‘impliceert’ een lichtere dekkingsvorm, dat wil zeggen: dekt minimaal alle testsituaties van de lichtere dekkingsvorm af.

Condition coverage	De mogelijke uitkomsten (“waar” of “onwaar”) van elke conditie worden minimaal één keer getest.
Decision coverage	De mogelijke uitkomsten van de beslissing worden minimaal één keer getest.
Condition/decision coverage	De mogelijke uitkomsten van elke conditie én van de beslissing worden minimaal één keer getest. Dit impliceert zowel “condition coverage” als “decision coverage”.
Modified condition/decision coverage	Elke mogelijke uitkomst van een conditie is minimaal één keer bepalend voor de uitkomst van de beslissing. Dit impliceert “condition/decision coverage”.
Multiple condition coverage	Alle mogelijke combinaties van uitkomsten van condities in een beslissing (dus de volledige beslistabel) worden minimaal één keer getest. Dit impliceert “modified condition/decision coverage”.

Tabel 14.2: Overzicht van de bekendste dekkingsvormen m.b.t. beslispunten.

Onderstaand voorbeeld illustreert de verschillen tussen de genoemde dekkingsvormen.

Uitgediept

Neem als voorbeeld het volgende beslispunt:

ALS (aantal boeken > 8 OF bedrag ≥ €250) DAN extra korting

Onderstaande tabellen laten dan zien met welke situaties de betreffende dekkingsvorm gehaald wordt.

Condition coverage

aantal boeken > 8	bedrag ≥ €250	Uitkomst
1	0	1 (extra korting)
0	1	1 (extra korting)

Decision coverage

aantal boeken > 8	bedrag ≥ €250	Uitkomst
0	1	1 (extra korting)
0	0	0

Condition/decision coverage

aantal boeken > 8	bedrag ≥ €250	Uitkomst
1	1	1 (extra korting)
0	0	0

Modified condition/decision coverage (zie volgende paragraaf)

aantal boeken > 8	bedrag ≥ €250	Uitkomst
1	0	1 (extra korting)
0	1	1 (extra korting)
0	0	0

Multiple condition coverage

aantal boeken > 8	bedrag ≥ €250	Uitkomst
1	1	1 (extra korting)
1	0	1 (extra korting)
0	1	1 (extra korting)
0	0	0

Bij sommige dekkingsvormen zijn er andere keuzes van situaties mogelijk die tot hetzelfde doel leiden. Zo wordt condition coverage ook bereikt met de twee situaties “1 1” en “0 0”. En decision coverage ook met “1 0” en “0 0” (en overigens ook met “1 1” en “0 0”).

Als lichtste dekkingsvorm kan het beste gekozen worden voor “condition/decision coverage”. Deze dekkingsvorm vervult zowel “condition coverage” als “decision coverage” en vereist hetzelfde aantal testsituaties.

De dekkingsvorm “modified condition/decision coverage” is een zeer krachtig middel om met een gering aantal testsituaties toch een grondige dekking te bereiken. Deze dekkingsvorm blijkt niet voor iedereen direct te doorgronden.

Daarom wordt deze in de volgende paragraaf uitvoerig verder toegelicht.

Modified condition/decision coverage

Definitie

Modified condition/decision coverage (afgekort MCDC) is de dekkingsvorm die het volgende garandeert:
Elke mogelijke uitkomst van een conditie is minimaal één keer bepalend voor de uitkomst van de beslissing.

Een belangrijk begrip in deze definitie is “bepalend”. Als de uitkomst van de conditie verandert (van 0 naar 1 of omgekeerd) dan verandert daarmee de uitkomst van het gehele beslispunt. De betekenis van MCDC kan als volgt toegelicht worden:

Als een beslispunt bestaat uit de condities A, B en C, dan garandeert MCDC

- dat er minstens één testsituatie is waarbij de uitkomst WAAR is, dankzij het feit dat conditie A WAAR is;
- dat er minstens één testsituatie is waarbij de uitkomst ONWAAR is, dankzij het feit dat conditie A ONWAAR is;
- idem voor conditie B en idem voor conditie C.

Dit is een grondige dekking, waarmee bijvoorbeeld de volgende fouten in het te testen systeem opgespoord zouden worden:

- Er is een conditie ten onrechte niet aanwezig;
- De “EN” is ten onrechte geïmplementeerd als een “OF”, en andersom;
- Een conditie is ‘omgekeerd’ geïmplementeerd, zoals “<” in plaats van “>” of “≠” in plaats van “=”.

Het grote voordeel van deze dekkingsvorm is zijn efficiëntie: Bij een beslispunt dat uit N condities bestaat, zijn meestal slechts **N+1 testsituaties** nodig voor MCDC. Vergeleken met het maximale aantal testsituaties (de volledige beslistabel) van 2^N is dat een enorme reductie, met name als N groot is (dus voor complexe beslispunten). Deze combinatie van “grondige dekking” met “relatief weinig testsituaties” maakt deze dekkingsvorm een krachtig gereedschap in het arsenaal van de tester.

Volgens de definitie van MCDC moet iedere conditie een keer *bepalend* zijn voor de uitkomst van de beslissing. Dan moeten alle andere condities in die situatie een waarde krijgen die *geen invloed* heeft op de uitkomst van de beslissing. Deze waarde wordt de “neutrale waarde” genoemd en wordt hieronder verder toegelicht.

Uitgediept

Neem bijvoorbeeld de complexe beslissing (R) die bestaat uit een samenstel van twee condities (A; B). De uitkomst van de beslissing is pas waar als beide condities waar zijn. Met andere woorden:

R = A EN B

Stel nu dat conditie B ONWAAR (of waarde 0) is. Dan maakt het niet meer uit wat A is, de uitkomst R is altijd ONWAAR. Anders gezegd:

A EN 0 = 0

Als echter conditie B WAAR (of waarde 1) is, dan hangt de uitkomst R van de beslissing geheel af van de uitkomst van A. Anders gezegd:

A EN 1 = A

Dit laatste is nu precies wat bereikt moet worden bij MCDC:

Als conditie A bepalend moet zijn voor de uitkomst van de beslissing (bij deze 'EN-samenstelling'), dan moet de andere conditie op waarde 1 ingesteld worden. We spreken hier over de **neutrale waarde**, omdat de waarde 1 geen invloed heeft op de einduitkomst.

Op dezelfde manier kan gesproken worden over de neutrale waarde voor de 'OF-samenstelling', die in dit geval 0 is. Immers, er geldt:

A OF 0 = A

Samengevat:
De neutrale waarde bij EN = 1
De neutrale waarde bij OF = 0

6-stappenplan voor het afleiden van testsituaties voor MCDC

Gebruikmakend van het begrip "neutrale waarde", kan het bereiken van MCDC nu concreet vertaald worden naar de eisen:

- Laat iedere conditie binnen de beslissing eenmaal WAAR en eenmaal ONWAAR zijn,
- waarbij alle *overige* condities in die testsituatie een *neutrale waarde* krijgen.

Er bestaan verschillende manieren om de benodigde testsituaties af te leiden.

Het 6-stappenplan is een techniek die zeer direct aansluit op de definitie van MCDC en waarmee op eenvoudige wijze een tabel ontstaat met alle benodigde testsituaties. Het 6-stappenplan wordt hieronder toegelicht met een relatief eenvoudig voorbeeld. Daarna wordt toegelicht hoe deze techniek werkt voor complexere samenstellingen van condities.

Neem als voorbeeld het volgende beslispunt dat beschrijft dat goede klanten extra korting krijgen:

ALS (aantal boeken > 8 OF bedrag ≥ €250) DAN extra korting

Het beslispunt is hier samengesteld uit twee condities met de structuur: R = A OF B.

Hieronder wordt het 6-stappenplan uitgewerkt die de testsituaties levert waarmee dit beslispunt gedekt wordt met MCDC.

1. Maak een tabel met drie kolommen.
   Vul de eerste rij als volgt:

R = A OF B	1	0

2. Voeg één rij toe voor iedere conditie in de beslissing.
   Deze rij gaat de twee testsituaties bevatten waarin de betreffende conditie bepalend is voor de uitkomst van het beslispunt. De conditie is eenmaal bepalend voor de uitkomst "1" en eenmaal bepalend voor de uitkomst "0".
   Zet in de eerste kolom de beschrijving van iedere conditie.

R = A OF B	**1**	**0**
A: aantal boeken > 8		
B: bedrag ≥ €250		

3. Vul de overgebleven cellen in de tabel met een aantal puntjes gelijk aan het aantal condities in de beslissing.
   Iedere cel is een testsituatie, die aan moet geven welke combinatie van WAAR/ONWAAR voor de condities geldt.

R = A OF B	**1**	**0**
A: aantal boeken > 8	. .	. .
B: bedrag ≥ €250	. .	. .

4. Vul de diagonaal in de tweede kolom met “1” en in de derde kolom met “0”.
   Dit is feitelijk het invullen van de **bepalende waarden** voor iedere conditie.
   Immers, de betekenis van bijvoorbeeld de cel die hoort bij rij “A” en kolom “1” is: “Dit is de testsituatie waarbij conditie A bepalend is voor een uitkomst van 1.”

R = A OF B	**1**	**0**
A: aantal boeken > 8	1 .	0 .
B: bedrag ≥ €250	. 1	. 0

5. Vul op de overgebleven puntjes een **neutrale waarde** in.
   In dit geval geldt voor zowel A als B dat ze via een OF-samenstelling aan de uitkomst bijdragen. Dus voor beide geldt de neutrale waarde van 0.

R = A OF B	**1**	**0**
A: aantal boeken > 8	1 0	0 0
B: bedrag ≥ €250	0 1	0 0

6. Streep dubbele voorkomens van testsituaties weg.

R = A OF B	**1**	**0**
A: aantal boeken > 8	1 0	0 0
B: bedrag ≥ €250	0 1	~~0 0~~

Uitgediept

Iedere testsituatie kan verder uitgewerkt worden, door aan te geven welke eisen aan de parameters van het beslispunt gesteld zijn om in deze testsituatie te komen. Dit is met name handig als de tester te maken krijgt met veel beslispunten en daarmee met veel testsituaties die tot testgevallen gecombineerd moeten worden.

In het bovenstaande voorbeeld kunnen de drie testsituaties als volgt uitgewerkt worden:

Testsituatie	**10**	**01**	**00**
aantal boeken	> 8	≤ 8	≤ 8
bedrag	< €250	≥ €250	< €250

Het hierboven beschreven 6-stappenplan werkt voor iedere samengestelde conditie, hoe complex dan ook. Bij samengestelde condities waarbij zowel “EN” als “OF” voorkomen, moet men wel opletten bij stap 5 (het invullen van de neutrale waarden). Onderstaand voorbeeld zal dit toelichten.

Neem als voorbeeld het beslispunt R = (A OF B) EN C.
Na de eerste vier stappen ziet de tabel met testsituaties er als volgt uit:

R = (A OF B) EN C	**1**	**0**
A	1 . .	0 . .
B	. 1 .	. 0 .
C	. . 1	. . 0

Nu moeten de neutrale waarden ingevuld worden voor elk van de 6 situaties.

Bij de bovenste twee situaties (A is de beslissende waarde) kunnen de neutrale waarden direct bepaald worden: B is met A verbonden via de operator “OF” en moet dus de neutrale waarde “0” krijgen. C is verbonden met A via de operator “EN” en moet dus de neutrale waarde “1 krijgen” Hetzelfde geldt voor de middelste twee situaties (B is de beslissende waarde).

Echter voor de onderste situatie (C is de beslissende waarde) is een tussenstap nodig: Het is niet A die rechtstreeks met C verbonden is en ook niet B. Het is de combinatie “(A OF B)” die met C verbonden is en wel via de operator “EN”. Dus “(A OF B)” moet de neutrale waarde van “EN” aannemen, en dat is dus “1”. Met andere woorden: (A OF B) = 1. Voor de waarden voor A en B zijn drie mogelijkheden om dit te bereiken, namelijk “1 1”, “1 0” of “0 1”.

Er hoeft slechts één van de drie gekozen te worden om het doel van MCDC te bereiken. In principe maakt het niet uit welke. Het enige verschil in de drie mogelijkheden is, dat bij het kiezen van “1 0” of “0 1” een testsituatie weggestreept

kan worden, terwijl dat voor de keuze “1 1” niet kan.

Als gekozen wordt voor de neutrale waarde “1 0” voor (A OF B), dan ziet de tabel er na de zes stappen als volgt uit:

R = (A OF B) EN C	1	0
A	1 0 1	0 0 1
B	0 1 1	~~0 0 1~~
C	~~1 0 1~~	1 0 0

Dit fenomeen dat er meerdere mogelijkheden zijn voor neutrale waarden treedt *altijd* op in geval van een operator tussen haakjes.

14.3.4 Equivalentieklassen

Het dekken van equivalentieklassen is een krachtig middel om met een beperkte set testsituaties een relatief hoge foutvindkans te bereiken. Het principe is eenvoudig en wordt door de meeste ervaren testers vanzelfsprekend en intuïtief toegepast.

Definitie

Bij het toepassen van equivalentieklassen wordt het volledige waardebereik van een parameter opgedeeld (gepartitioneerd) in klassen waarbij het systeemgedrag gelijksoortig (equivalent) is.

Deze klassen worden equivalentieklassen genoemd. Andere (Engelse) termen waaronder het ontwerpen van testgevallen op basis van equivalentieklassen bekend staat, zijn “equivalence partitioning” en “domain testing”.

Uitgediept

Ter verduidelijking het volgende voorbeeld:

Bij het al of niet verstrekken van een lening, spelen vele parameters een rol. Eén van die parameters is de *leeftijd* van de lening-nemer. In het lening-systeem kan de leeftijd variëren tussen de 0 en de 100. Voor wat betreft de leeftijd zijn de regels voor het al of niet verstrekken van een lening als volgt:

jonger dan 18	Geen lening verstrekken
van 18 t/m 55	Standaard lening verstrekken
ouder dan 55	Duurdere lening verstrekken

Stel nu dat er drie testgevallen uitgevoerd worden, met achtereenvolgens de volgende waarden voor leeftijd: 16, 32 en 44. Dit is om meerdere redenen geen verstandige keuze:

- Er is geen enkel testgeval waarbij een duurdere lening verstrekt wordt. Een eventuele fout bij het verstrekken van duurdere leningen wordt dus niet gevonden.
- Als het systeem correct werkt bij leeftijd = 32, dan is het erg onwaarschijnlijk dat er een fout zal optreden bij leeftijd = 44. Het laatste testgeval is dus overbodig.

In dit voorbeeld bestaan voor de parameter “leeftijd” drie equivalentieklassen, namelijk:

0 t/m 17; 18 t/m 55; 56 t/m 100.

De verschillende soorten systeemgedrag met betrekking tot deze parameter worden al afgedekt met één testgeval uit iedere equivalentieklasse.

Het principe achter het toepassen van equivalentieklassen is, dat iedere waarde uit een dergelijke klasse dezelfde kans heeft op het vinden van een fout én dat het testen met meerdere waarden uit dezelfde klasse deze foutvindkans nauwelijks vergroot. Het is goed om te beseffen dat dit een *aanname* is. Als bij een willekeurige waarde in een equivalentieklasse een correct systeemgedrag optreedt, is het in principe nog steeds mogelijk dat er een fout optreedt bij een andere waarde.

Bijvoorbeeld:

- Frauduleus systeemgedrag.
  Bijvoorbeeld als een programmeur met frauduleuze bedoelingen in de code heeft toegevoegd: “ALS naam = [mijn naam] DAN korting = [belachelijk hoog]” Dit kan in principe op geen enkele gestructureerde wijze gevonden worden, alleen door puur toeval (of door het inspecteren van de broncode zelf).
- In de verkeerde equivalentieklasse terecht komen.
  In bepaalde situaties kan systeemgedrag optreden dat bij een andere equivalentieklasse hoort. Meestal heeft dat te maken met de grenswaarden die aan de verkeerde equivalentieklasse toegekend worden. Zie verder hierover paragraaf 14.3.6 “Grenswaardenanalyse”.

Ook al is het achterliggende principe een aanname, het is een bruikbare en nuttige aanname. Door testgevallen te baseren op deze equivalentieklassen in plaats van op elke mogelijke invoerwaarde, blijft het aantal testgevallen beperkt, terwijl toch een goede dekking verkregen wordt van de mogelijke variaties in het systeemgedrag.

Soms wordt onderscheid gemaakt tussen geldige en ongeldige equivalentieklassen: Invoerwaarden uit de ongeldige equivalentie klasse leiden tot foutmeldingen. Invoerwaarden uit een geldige equivalentieklasse worden verwerkt zoals bedoeld.

Deze speciale vorm van dekking valt onder het dekken van “Goedpaden / Foutpaden” (zie paragraaf 14.3.9).

Equivalentieklassen zijn toepasbaar op zowel invoerparameters als uitvoerparameters.

14.3.5 Orthogonale arrays en Pairwise testing

Orthogonale arrays is een tamelijk recente wiskundige theorie die steeds meer in de belangstelling komt. Het houdt zich bezig met de wiskundige eigenschappen van het combineren van factoren, de zogenaamde “combinatoriek”. Het heeft al tal van interessante toepassingen in wetenschap en industrie en … testen.

Ook bij testen wordt gebruik gemaakt van combinaties. Bedenk dat het systeemgedrag in het algemeen bepaald wordt door meerdere parameters die elk verschillende waarden kunnen aannemen. Voor het maken van een testgeval kiest de tester voor elke parameter één van de mogelijke waarden. Feitelijk creëert hij een *combinatie van parameterwaarden* die getest gaat worden. Het testen van alle mogelijke combinaties is in de praktijk al gauw onhaalbaar. Als er bijvoorbeeld sprake is van 8 parameters die elk 3 mogelijke waarden (of equivalentieklassen) hebben, dan zijn er al $3^8 = 6561$ mogelijkheden! Noodgedwongen zal er slechts een subset getest kunnen worden. maar wat is een verstandige subset? Hier kunnen orthogonale arrays en pairwise testing een oplossing bieden.

Dit hoofdstuk legt eerst het principe van pairwise testing uit, zodat de lezer bekend raakt met de bedoeling en het resultaat van dergelijke technieken die gebaseerd zijn op combinatoriek. Daarna wordt de complexere materie van orthogonale arrays op hoofdlijnen toegelicht. Dit laatste is niet noodzakelijk om de basistechniek van pairwise testing te kunnen toepassen, maar is voor de liefhebbers van verdere theoretische verdieping.

Pairwise testing

Pairwise testing gaat uit van het fenomeen dat de meeste fouten in software het gevolg zijn van één bepaalde factor of de combinatie van 2 factoren. Het aantal fouten dat veroorzaakt wordt door een specifieke combinatie van meer dan 2 factoren wordt exponentieel kleiner. In plaats van *alle* mogelijke combinaties van *alle* factoren te testen, is het al erg effectief om in ieder geval elke combinatie van 2 willekeurige factoren getest te hebben.

Definitie

Pairwise testing heeft tot doel om van elke willekeurige combinatie van 2 factoren alle mogelijkheden te testen.

Dit levert een enorme reductie in het aantal benodigde testgevallen met nog steeds een goed resultaat in het vinden van fouten.

Het volgende voorbeeld illustreert wat pairwise testing voorstelt.

Uitgediept

Van het te testen systeem (voor het bestellen van boeken via internet) spelen de volgende drie parameters een rol. Van elke parameter zijn 2 equivalentieklassen die getest moeten worden:

aantal boeken	weinig; veel
bedrag	laag; hoog
ledenkaart	geen; gold card

Om alle combinaties met betrekking tot deze drie parameters te testen, zijn 2x2x2=8 testgevallen nodig, namelijk:

	aantal boeken	bedrag	ledenkaart
1	weinig	laag	geen
2	weinig	laag	gold card
3	weinig	hoog	geen
4	weinig	hoog	gold card
5	veel	laag	geen
6	veel	laag	gold card
7	veel	hoog	geen
8	veel	hoog	gold card

Voor pairwise testing kan worden volstaan met slechts vier testgevallen, zoals hieronder weergegeven:

	aantal boeken	bedrag	ledenkaart
1	weinig	laag	geen
2	weinig	hoog	gold card
3	veel	laag	gold card
4	veel	hoog	geen

Van de twee parameters [aantal boeken, bedrag] worden alle vier bestaande combinaties getest (weinig/laag; weinig/hoog; veel/laag; veel/hoog). Datzelfde geldt ook voor de andere combinaties van twee parameters, dus voor [aantal boeken, ledenkaart] en [bedrag, ledenkaart]. Ga dit zelf na.

Wat is nu het nut hiervan? Als er een fout bestaat in het systeem die optreedt zodra één van de mogelijke waarden van één van de parameters gecombineerd wordt met een bepaalde waarde van één van de andere parameters, dan wordt deze fout *altijd* gevonden met deze 4 testgevallen. Dat is de kracht van pairwise testing.

Afleiden van testgevallen voor pairwise testing

Als er sprake is van enkele parameters met slechts 2 equivalentieklassen, kunnen de benodigde testgevallen voor pairwise testing nog wel handmatig afgeleid worden. In de meeste praktijksituaties is echter sprake van meer parameters en equivalentieklassen en is dit niet meer handmatig te doen. Dan zijn er twee manieren om dit op te lossen:

- Toepassen van orthogonale arrays (zie volgende paragraaf);
- Gebruik van tools (wordt in deze paragraaf verder uitgewerkt).

Het aantal benodigde testgevallen en hoe ze precies samengesteld zijn, hangt af van de tool die gebruikt wordt. Er zijn diverse tools beschikbaar, zowel commercieel als gratis via internet (freeware). In het voorbeeld hieronder is het resultaat beschreven van twee freeware tools.

Neem bijvoorbeeld de volgende uitgebreidere versie van het “boek-bestel-systeem”. De eerder genoemde parameters hebben nu meer equivalentieklassen en er is een parameter “bestelseizoen” bijgekomen:

aantal boeken	1; 2-8; >8
bedrag	<100; 100-250; >250
ledenkaart	geen; zilver; goud; platinum
bestelseizoen	doordeweeks; weekend; feestdag

Voor het testen van alle mogelijke combinaties zijn in dit geval 3x3x4x3=108 testgevallen nodig. Voor pairwise testing

kan volstaan worden met een fractie hiervan. Hieronder worden de resultaten getoond die met twee freeware tools behaald zijn:

Resultaten met tool "Pict33"
Dit is een tool van Microsoft® en kan gedownload worden via hun site of via www.pairwise.org. Deze tool genereert de volgende 14 testgevallen:

	aantal boeken	bedrag	ledenkaart	bestelseizoen
1	1	<100	geen	feestdag
2	1	>250	platinum	doordeweeks
3	1	>250	zilver	doordeweeks
4	1	100-250	goud	weekend
5	>8	<100	zilver	weekend
6	>8	>250	geen	doordeweeks
7	>8	>250	goud	feestdag
8	>8	100-250	platinum	feestdag
9	>8	100-250	platinum	doordeweeks
10	2-8	<100	platinum	doordeweeks
11	2-8	<100	goud	doordeweeks
12	2-8	>250	platinum	weekend
13	2-8	100-250	zilver	feestdag
14	2-8	100-250	geen	weekend

Resultaten met tool "Allpairs"
De maker van dit tool is James Bach. Het is te downloaden van www.satisfice.com. Deze tool genereert de volgende 13 testgevallen:

	aantal boeken	bedrag	ledenkaart	bestelseizoen
1	1	<100	geen	doordeweeks
2	2-8	100-250	geen	weekend
3	>8	>250	geen	feestdag
4	1	100-250	zilver	feestdag
5	2-8	<100	zilver	doordeweeks
6	>8	<100	zilver	weekend
7	1	>250	goud	weekend
8	2-8	<100	goud	feestdag
9	>8	100-250	goud	doordeweeks
10	2-8	>250	platinum	doordeweeks
11	1	<100	platinum	weekend
12	>8	100-250	platinum	feestdag
13	~1	>250	zilver	~doordeweeks

Het teken "~" in het laatste testgeval betekent, dat de waarde er hier niet toe doet. Er had ook een andere equivalentieklasse gekozen mogen worden. Het testgeval is uitsluitend nodig om de dekking van de andere parameters te completeren.

Zwaardere varianten zoals triplewise testing

Het principe van pairwise testing kan uitgebreid worden tot het combineren van meer dan twee factoren. Zo spreekt men bij het testen van alle mogelijkheden van iedere **drie** willekeurige factoren over triplewise testing. Algemeen kan gedefinieerd worden:

Definitie

N-wise testing heeft tot doel om van elke willekeurige combinatie van N factoren alle mogelijkheden te testen.

De maximale waarde voor N is gelijk aan het aantal parameters. In dat geval is het resultaat gelijk aan het testen van de volledige beslistabel: alle combinaties van alle waarden van alle parameters. In de praktijk wordt een waarde van 4 of hoger zelden toegepast.

De eerder genoemde tool "Pict33" kan een willekeurig opgegeven zwaarte aan. Voor triplewise testing van het hiervoor genoemde voorbeeld, berekent deze tool 43 testsituaties. Toepassen van N=4 zou het maximale aantal van 108

testsituaties opleveren.

Orthogonale arrays

In het eerste voorbeeld van pairwise testing werd gesproken over drie parameters met elk 2 equivalentieklassen. Als bij iedere parameter de equivalentieklassen genummerd worden, dus “1” en “2”, dan worden de 4 testgevallen voor pairwise testing beschreven door de volgende 2-dimensionale array:

	param-1	**param-2**	**param-3**
TC1	1	1	1
TC2	1	2	2
TC3	2	1	2
TC4	2	2	1

Dit is een voorbeeld van een orthogonale array en wordt genoteerd als $L_4(2^3)$.

De betekenis van de verschillende symbolen is hier:

4 = het aantal rijen (dus het aantal testgevallen dat benodigd blijkt te zijn);
3 = het aantal kolommen (dus de parameters die tezamen een testgeval vormen);
2 = de maximale waarde in iedere kolom (dus het aantal equivalentieklassen van iedere parameter).

De interessante eigenschap van deze array is, dat iedere combinatie van 2 kolommen alle mogelijke combinaties van de waarden “1” en “2” bevat. De term hiervoor is dat het een orthogonale array van “**strength 2**” is. Dit sluit precies aan bij het doel van pairwise testing, waarbij “alle combinaties van equivalentieklassen van TWEE willekeurige parameters” getest wordt.

Een andere bijzondere eigenschap van orthogonale arrays is, dat ze **gebalanceerd** zijn. Dat wil zeggen dat iedere combinatie van parameterwaarden *even vaak* voorkomt als iedere andere combinatie van parameterwaarden. Deze eigenschap onderscheidt orthogonale arrays van pairwise testing. Bij de testgevallen die ontstaan door het toepassen van pairwise testing is het gebruikelijk dat bepaalde equivalentieklassen vaker voorkomen dan andere. Bij orthogonale arrays komt iedere equivalentieklasse (zelfs ieder combinatie van equivalentieklassen) even vaak voor.

Voor het zwaarder testen dan pairwise testing is een orthogonale array met een hogere strength nodig. Zo zal men voor het testen van alle combinaties van equivalentieklassen van DRIE willekeurige parameters een orthogonale array van “**strength 3**” nodig hebben. Hieronder staat een orthogonale array van strength 3, voor 4 parameters die elk 2 waarden kunnen hebben:

1 1 1 1
1 1 2 2
1 2 1 2
1 2 2 1
2 1 1 2
2 1 2 1
2 2 1 1
2 2 2 2

Deze array heeft de eigenschap dat iedere combinatie van 3 kolommen alle mogelijke combinaties van de waarden 1 en 2 bevat, namelijk (1 1 1), (1 1 2), (1 2 1), (1 2 2), (2 1 1), (2 1 2), (2 2 1) en (2 2 2).

De notatie voor deze orthogonale array is: $L_8(2^4, 3)$, waarbij het getal 3 dus de strength van de array aangeeft. In de literatuur bestaan verschillende manieren om orthogonale arrays te noteren. De meest gebruikte notatiewijzen zijn:

- $L_8(2^4, 3)$,
- $OA(8, 2^4, 3)$,
- OA(8, 4, 2, 3)

Als de waarde voor strength weggelaten is, wordt de default waarde strength 2 aangenomen.

Al het voorgaande kan samengevat worden in de volgende ingewikkelde definitie:

Definitie

Een orthogonale array $L_N(s^k, t)$ is een 2-dimensionale array van N rijen en k kolommen bestaande uit elementen die s waarden kunnen aannemen, waarbij iedere combinatie van t kolommen alle combinaties van de s waarden in gelijke hoeveelheid bevat.

Niet voor iedere waarde van "s" en "k" bestaat een orthogonale array. In de praktijk wordt in zo'n situatie een 'te grote' array gebruikt, waarbij de overbodige kolommen bij de toepassing ervan genegeerd worden.

Tot nu toe zijn er steeds orthogonale arrays beschreven waarbij de mogelijke waarden voor de elementen van alle kolommen gelijk is. Er bestaan ook orthogonale arrays waarbij kolommen verschillende mogelijke waarden mogen hebben. Een voorbeeld hiervan is $L_8(2^4, 4^1, 2)$. Deze array van strength 2 bevat 4 kolommen die elk 2 mogelijke waarden hebben plus één kolom met 4 mogelijke waarden. Hieronder staat zo'n array uitgewerkt:

0	0	0	0	0
1	1	1	1	0
0	0	1	1	1
1	1	0	0	1
0	1	0	1	2
1	0	1	0	2
0	1	1	0	3
1	0	0	1	3

Hoe kom je aan orthogonale arrays?

Het creëren van een orthogonale array is niet eenvoudig. Als er sprake is van meerdere kolommen die meer dan twee waarden kunnen aannemen, is het al vrijwel niet meer handmatig te doen. Gelukkig is er een uitgebreide literatuur, zowel boeken als artikelen als internet-sites, over orthogonale arrays, met vele voorbeelden en uitgewerkte arrays waar de tester gebruik van kan maken.

Het boek "Orthogonal arrays: Theory and Applications" [Hedayat, 1999] wordt meestal gezien als standaardwerk over orthogonale arrays.

Eén van de meest uitgebreide verzamelingen orthogonale arrays op internet is te vinden op de site van dr. N.J.A Sloane. Een internet-search op "orthogonal arrays" zal ongetwijfeld naar deze en vele andere bruikbare sites leiden.

Afleiden van testgevallen met orthogonale arrays

Het toepassen van pairwise testing (en zwaardere varianten) is eenvoudig dankzij de beschikbare tools die snel de benodigde testgevallen genereren. Het afleiden van de testgevallen door gebruik te maken van orthogonale arrays is meer werk en ingewikkelder, maar geeft het extra voordeel van de "gebalanceerde testgevallen": iedere combinatie van equivalentieklassen komt even vaak voor. De tester zal moeten afwegen of dit extra voordeel opweegt tegen de extra inspanning die benodigd is.

Voor de liefhebbers wordt hieronder toegelicht hoe orthogonale arrays toegepast kunnen worden om een gebalanceerde set testgevallen voor N-wise testing af te leiden:

1. Bepaal de parameters waaruit het testgeval bestaat.
2. Bepaal de equivalentieklassen van iedere parameter.
3. Kies de strength van de orthogonale array, gebaseerd op de gewenste zwaarte van de test.
4. Zoek een passende orthogonale array.
   Niet voor iedere hoeveelheid parameters en equivalentieklassen bestaat een orthogonale array. In dat geval moet er gezocht worden naar één die 'net iets te groot' is. Dit kan op twee manieren en moet dan als volgt gebruikt worden:
   - Er zijn te veel kolommen. (Meer kolommen dan parameters.)

De betreffende kolommen kunnen gewoon weggelaten worden. Dit heeft geen invloed op de orthogonaliteit van de array.
- De maximale waarde van een kolom is te groot. (De maximale waarde “s” is groter dan het aantal equivalentieklassen van de parameter.)

Hier mogen niet zomaar de rijen die de te grote waarde bevatten, weggelaten worden. Deze rijen zijn nodig om de verplichte combinaties van de andere parameters te realiseren. In dit geval moet een te grote waarde in de kolom opgevat worden als “de waarde doet er niet toe”. Er kan dus een willekeurige equivalentieklasse gekozen worden.

5. Vertaal de orthogonale array naar testgevallen.

Voorbeeld

Om dit toe te lichten wordt hetzelfde voorbeeld als bij pairwise testing uitgewerkt, over het “boek-bestel-systeem”: Stel dat dit systeem gemiddeld zwaar getest moet worden. De vijf stappen voor het afleiden van de testgevallen lopen dan als volgt:

Stap 1: Bepaal de parameters waaruit het testgeval bestaat

Er zijn vier parameters: “aantal boeken”, “bedrag”, “ledenkaart”, “bestelseizoen”.

Stap 2: Bepaal de equivalentieklassen van iedere parameter

De parameters zijn oplopend gesorteerd naar het aantal equivalentieklassen.

Dit maakt het zoeken naar een geschikte orthogonale array (stap 4) eenvoudiger.

aantal boeken	1; 2-8; >8
bedrag	<100; 100-250; >250
bestelseizoen	doordeweeks; weekend; feestdag
ledenkaart	geen; zilver; goud; platinum

Stap 3: Kies de strength van de orthogonale array

Omdat het systeem gemiddeld zwaar getest moet worden, wordt gekozen voor strength 2. Voor nog zwaarder testen, kan een hogere strength gekozen worden. Dit heeft voornamelijk effect op het zoeken van een passende orthogonale array (stap 4). Verder blijven de vijf stappen in deze techniek onveranderd.

Stap 4: Zoek een passende orthogonale array

Er zijn 3 parameters met 3 equivalentieklassen en 1 parameter met 4 equivalentieklassen. Dat wil zeggen dat er gezocht moet worden naar een orthogonale array die zo dicht mogelijk komt bij

$L_N (3^3, 4^1, 2)$

Helaas bestaat deze orthogonale array niet. Dus er moet gezocht worden naar eentje die net iets groter is. De volgende twee orthogonale arrays zouden in aanmerking komen:

- $L_{18}(3^6, 6^1, 2)$
  Deze bestaat uit 18 rijen en 7 kolommen. Van de eerste 6 kolommen (die 3 waarden kunnen aannemen) zijn er 3 overbodig en kunnen geschrapt worden. De laatste kolom kan 6 waarden aannemen, maar daarvan zijn er voor onze test slechts 4 nodig. Deze kolom heeft dus 2 overbodige waarden die willekeurig ingevuld mogen worden (de “doet-er-niet-toe waarden”).
- $L_{16}(4^5, 2)$
  Deze bestaat uit 16 rijen en 5 kolommen. Eén kolom is overbodig en kan geschrapt worden. De eerste 3 kolommen van de overgebleven tabel bevatten dan nog één mogelijke waarde te veel, die dus als “doet-er-niet-toe waarde” geldt.

Er wordt gekozen voor de tweede array, omdat die het kleinste aantal benodigde testgevallen levert. De $L_{16}(4^5, 2)$ ziet er als volgt uit:

1	1	1	1	1
1	2	2	2	2
1	3	3	3	3
1	4	4	4	4
2	1	2	3	4
2	2	1	4	3
2	3	4	1	2
2	4	3	2	1
3	1	3	4	2
3	2	4	3	1
3	3	1	2	4
3	4	2	1	3
4	1	4	2	3
4	2	3	1	4
4	3	2	4	1
4	4	1	3	2

Stap 5: Vertaal de orthogonale array naar testgevallen

De gekozen orthogonale array wordt nu op maat gesneden en vervolgens ingevuld met de echte waarden van de test-parameters.

Van de array wordt de laatste kolom verwijderd. De eerste 3 parameters hebben slechts 3 mogelijke waarden. Daarom wordt van de eerste 3 kolommen de waarde “4” vervangen door een “~” die aangeeft dat de waarde er niet toe doet. De array ziet er dan als volgt uit:

1	1	1	1
1	2	2	2
1	3	3	3
1	~	~	4
2	1	2	3
2	2	1	4
2	3	~	1
2	~	3	2
3	1	3	4
3	2	~	3
3	3	1	2
3	~	2	1
~	1	~	2
~	2	3	1
~	3	2	4
~	~	1	3

Uitgediept

Met een slimme keuze van de “doet-er-niet-toe waarden” kan een aantal identieke rijen gecreëerd worden: Voor de rijen 4 en 15 kan de identieke combinatie (1 3 2 4) gekozen worden en voor de rijen 10 en 16 de combinatie (3 2 1 3). Hiermee vervallen dan twee rijen en dus twee testgevallen. Deze reductie is echter niet verplicht en voor de verdere uitwerking zal dit buiten beschouwing gelaten worden.

Tenslotte moeten de testparameters toegekend worden aan de kolommen van de array en moeten de waarden in de kolommen vervangen worden door een equivalentieklasse. Welke waarde gekoppeld wordt aan welke equivalentieklasse, doet er niet toe. Als het maar consequent toegepast wordt. Ter herinnering staan hieronder nogmaals de parameters met hun equivalentieklassen opgesomd.

aantal boeken	1; 2-8; >8
bedrag	<100; 100-250; >250
bestelseizoen	doordeweeks; weekend; feestdag
ledenkaart	geen; zilver; goud; platinum

In ons voorbeeld komt het eindresultaat er dan als volgt uit te zien. Met een "~" wordt nog steeds bedoeld dat een willekeurige keuze uit de beschikbare waarden gemaakt mag worden.

	aantal boeken	bedrag	bestelseizoen	ledenkaart
1	1	<100	doordeweeks	geen
2	1	100-250	weekend	zilver
3	1	>250	feestdag	goud
4	1	~	~	platinum
5	2-8	<100	weekend	goud
6	2-8	100-250	doordeweeks	platinum
7	2-8	>250	~	geen
8	2-8	~	feestdag	zilver
9	>8	<100	feestdag	platinum
10	>8	100-250	~	goud
11	>8	>250	doordeweeks	zilver
12	>8	~	weekend	geen
13	~	<100	~	zilver
14	~	100-250	feestdag	geen
15	~	>250	weekend	platinum
16	~	~	doordeweeks	goud

14.3.6 Grenswaardenanalyse

Als het systeemgedrag verandert zodra de waarde van een parameter een bepaalde grens overschrijdt, is er sprake van een zogenaamde grenswaarde. Het volgende voorbeeld laat dit zien:

Voorbeeld

"Voor wat betreft de leeftijd zijn de regels voor het al of niet verstrekken van een lening als volgt

jonger dan 18	Geen lening verstrekken
van 18 t/m 55	Standaard lening verstrekken
ouder dan 55	Duurdere lening verstrekken

In dit voorbeeld zijn er drie equivalentieklassen voor de parameter leeftijd.

De waarden 18 en 55 zijn de grenswaarden voor deze parameter. Beide grenswaarden horen in dit geval bij de middelste equivalentieklasse "standaard lening", zoals geïllustreerd in figuur 14.8.

Figuur 14.8: Voorbeeld van twee grenswaarden en hun positie in de equivalentieklassen van de parameter "leeftijd".

In de praktijk blijken veel fouten te maken te hebben met grenswaarden.

Meestal zijn het gewoon 'slordige programmeerfouten' waarbij de programmeur bijvoorbeeld per ongeluk een ">" in plaats van een "≥" programmeerde, of een "=" in plaats van een "≥". Zo zou in bovenstaand voorbeeld ergens in de code per ongeluk de conditie "ALS leeftijd ≤ 18 DAN …" geprogrammeerd kunnen zijn. In dat geval zou de leeftijd 18 plotseling bij de eerste equivalentieklasse ("geen lening") ingedeeld zijn in plaats van bij de middelste ("standaard lening").
Behalve bij equivalentieklassen treden grenswaarden ook vaak op bij het dekken van condities en beslispunten. Zo zou bijvoorbeeld in het leensysteem de volgende conditie gedefinieerd kunnen zijn:

```
ALS ( leenbedrag > salaris ) DAN …
```

Hier is "leenbedrag" de parameter met als grenswaarde het "salaris".
Het testen of de grenswaarden wel bij de juiste equivalentieklasse (of uitkomst van de conditie) ingedeeld zijn, is een apart testdoel dat bereikt wordt met de zogenaamde "**grenswaardenanalyse**".
De techniek voor het uitvoeren van grenswaardenanalyse is uiterst eenvoudig:

- Bepaal de grenswaarden bij de betreffende equivalentieklasse of conditie;
- Definieer de volgende **drie testsituaties**: precies op de grens, direct erboven, direct eronder.

Wat precies de betekenis is van "direct erboven" of "direct eronder", hangt af van de eenheid en nauwkeurigheid waarmee de waarde van die parameter uitgedrukt wordt. Als het salaris en leenbedrag uitgedrukt worden in centen, dan moeten in eerder genoemd voorbeeld de volgende leenbedragen getest worden:

- leenbedrag == salaris
- leenbedrag == salaris + € 0,01
- leenbedrag == salaris – € 0,01

Als het systeem echter in 4 cijfers achter de komma rekent, moet in plaats van € 0,01 getest worden met een verschil van € 0,0001.

Er is ook een lichte variant van grenswaardenanalyse, waarbij slechts twee testsituaties getest worden: de grenswaarde zelf plus de daaraan grenzende waarde in de *andere* equivalentieklasse. Dit kan geïllustreerd worden met het voorbeeld aan het begin van deze paragraaf (zie figuur 14.8). De parameter leeftijd heeft drie equivalentieklassen en twee grenswaarden (18 en 55). Het aantal testsituaties bij geen-, normale- en lichte grenswaardenanalyse is dan als volgt:

- geen grenswaardenanalyse -> 3 testsituaties: bijv. 12, 32, 64
- normale grenswaardenanalyse -> 6 testsituaties: 17, 18, 19 en 54, 55, 56
- lichte grenswaardenanalyse -> 4 testsituaties: 17, 18 en 55, 56

Als er niet expliciet gekozen is voor grenswaardenanalyse, passen ervaren testers vaak toch intuïtief de lichte variant toe. Immers, als het toch vereist is om een waarde te testen uit beide equivalentieklassen (boven en onder de grenswaarde), kan zonder extra moeite gekozen worden voor "precies op" en "vlak naast" de grenswaarde.

Een nadeel van de lichte variant is, dat deze bepaalde fouten niet ontdekt die met de standaard grenswaardenanalyse wel gevonden worden. Een voorbeeld hiervan is de eerder genoemde fout, waarbij een "=" in plaats van een "≥" geprogrammeerd is.

Grenswaardenanalyse is *niet altijd toepasbaar* op equivalentieklassen of condities. Er hoeft namelijk niet altijd sprake te zijn van grenswaarden. Neem bijvoorbeeld de parameter "geslacht" met als waarden (en dus equivalentieklassen) "M" en "V". Er is niet zoiets als een grens tussen de "M" en de "V". Dit geldt bijvoorbeeld ook voor al die parameters in een systeem die behoren tot 'codes' en 'types'.

Grenswaardenanalyse is niet alleen van toepassing op de *invoerkant* van een systeem, maar ook op de *uitvoerkant*. Stel dat een offertepagina maximaal 10 regels mag bevatten. Dan wordt dit met grenswaardenanalyse getest door een offerte af te drukken met 9 regels, met 10 regels (alle regels op één pagina) en met 11 regels (elfde regel op tweede pagina).

14.3.7 CRUD

De gegevens die in het te testen systeem opgeslagen en onderhouden worden, kennen een *levensloop*. Deze begint als een gegeven gecreëerd wordt en eindigt wanneer het gegeven weer wordt verwijderd. Daartussen wordt het gegeven gebruikt

door het te wijzigen of te raadplegen.

Een overzicht van de levensloop van de gegevens c.q. entiteiten wordt verkregen met behulp van een zogenaamde *CRUD-matrix*. Dit is een matrix waarbij op de assen horizontaal de entiteiten en verticaal de functies staan weergegeven. Door middel van de letters C(reate), R(ead), U(pdate) en D(elete) wordt de matrix ingevuld. Als een functie een bepaalde actie met betrekking tot een entiteit uitvoert wordt dit in de matrix aangegeven door middel van een C, R, U en/of D. Dit is geïllustreerd in figuur 14.9.

	Factuur	**Artikel**	**…**
Beheren artikelen	-	C, U, D	…
Aanmaken factuur	C	R	…
Balie-betaling	C, R, D	-	…
…	…	…	…

Figuur 14.9: Voorbeeld van een CRUD-matrix.

Samenstellen CRUD-matrix

- Voor het samenstellen van de CRUD-matrix worden *alle* functies in het systeem doorlopen. Per onderkende functie wordt bepaald
- welke entiteiten door deze functie gebruikt worden;
  Het resultaat hiervan wordt in de matrix ingevuld.

Uitgediept

Vaak is een speciale structuur zichtbaar die te maken heeft met twee groepen van gegevens en van functies:

- Stamgegevens met beheerfuncties.
  Stamgegevens zijn de basisgegevens in het systeem. Bijvoorbeeld “artikel” en “klant”. Zij worden meestal onafhankelijk van de andere gegevens onderhouden met behulp van de hieraan gekoppelde beheerfuncties.
  Dit heeft in het algemeen het volgende effect op de CRUD-matrix: Bij een beheerfunctie is alleen de kolom van het betreffende stamgegeven ingevuld, en wel met alle acties: C, R, U en D. Als de stamgegevens en beheerfuncties als eerste (en in dezelfde volgorde) in de CRUD-matrix gedefinieerd zijn, zal dit deel van de CRUD-matrix uitsluitend op de diagonaal ingevuld zijn (met “CRUD”).
- Afgeleide gegevens met verwerkingsfuncties
  Afgeleide gegevens zijn gegevens die door de speci fieke bedrijfsprocessen geproduceerd worden, waarbij gebruik gemaakt wordt van stamgegevens. Bijvoorbeeld “offerte” en “factuur”. Het zijn de verwerkingsfuncties die de specifieke bedrijfsprocessen realiseren en de afgeleide gegevens produceren en wijzigen.
  Dit heeft in het algemeen het volgende effect op de CRUD-matrix: Verwerkings functies voeren alleen de actie “R” uit op stamgegevens. Op de afgeleide gegevens kunnen alle acties uitgevoerd worden. Bij de afgeleide gegevens zijn de rijen van de beheerfuncties leeg. Alle acties (C, R, U en D) worden door één of meer verwerkingsfuncties uitgevoerd.

Als hiervan in de praktijk afgeweken wordt, is dat natuurlijk toegestaan, maar dat is op z’n minst aanleiding om de reden ervan te onderzoeken.

Bij voorkeur wordt de CRUD-matrix niet pas tijdens het testtraject gemaakt, maar wordt hij als onderdeel van systeemontwikkeling opgeleverd door de ontwikkelaar. Het opstellen van een CRUD-matrix is namelijk niet alleen bruikbaar voor testers, maar ook voor de ontwikkelaars zelf: Bij het ontwerpen van een informatiesysteem wordt veelal vanuit de functies geredeneerd. Per functie is beschreven welke gegevens worden gebruikt. Bij het opstellen van een CRUD-matrix wordt vanuit de gegevens geredeneerd. Per entiteit wordt beschreven welke functies op welke wijze de betreffende entiteit gebruiken. Door het opstellen van een dergelijke cross-reference tabel (CRUDmatrix) komen soms onduidelijkheden en/of onvolledigheden aan het licht die bij een functiegerichte benadering waarschijnlijk niet worden gevonden en nu reeds in een vroegtijdig stadium worden ontdekt!

Testen van de levensloop

Het testen van de levensloop bestaat uit twee delen: volledigheidscheck en consistentietest. Dit wordt hieronder verder

toegelicht:

Volledigheidscheck

Dit is een *statische* test, waarbij in de CRUD-matrix onderzocht wordt of alle vier mogelijke acties (C, R, U en D) bij iedere entiteit voorkomen. Met andere woorden: Is voor iedere entiteit de volledige levensloop geïmplementeerd. Het ontbreken van een actie hoeft niet altijd te betekenen dat het systeem fout is. Wel is dat op zijn minst aanleiding om de reden ervan te onderzoeken.

Consistentietest

Dit is een *dynamische* test die zich richt op de integratie van de verschillende functies. Hierbij wordt gecontroleerd of de verschillende functies op een consistente manier een gegeven gebruiken. Met andere woorden: Wordt het betreffende gegeven niet door de ene functie zodanig gecorrumpeerd dat het door de andere functies niet meer correct gebruikt kan worden?

Testgevallen worden afgeleid door een volledige levensloop van een gegeven samen te stellen. Dit gebeurt op de volgende wijze:

- Ieder testgeval begint met een “C”, gevolgd door alle mogelijke “U”s en afgesloten met een “D”. Als er meerdere mogelijkheden zijn om een gegeven te creëren of te verwijderen, worden hiervoor meerdere testgevallen ontworpen.
- Na iedere actie (C, U of D) wordt één of meer keren een “R” uitgevoerd.
  Dit dient om vast te stellen dat het gegeven correct bewerkt is en bruikbaar is voor andere functies (niet corrupt geraakt).
- Van het betreffende gegeven moeten alle voorkomens van acties (C, R, U en D) bij alle functies door de testgevallen gedekt zijn.

Hiermee is in principe CRUD volledig gedekt.

Een **zwaardere dekking** van CRUD kan gerealiseerd worden door te eisen dat ook combinaties van acties volledig gedekt zijn. Bijvoorbeeld door te eisen dat na iedere “U” *alle* functies met een “R” uitgevoerd moeten worden.

Onderstaand voorbeeld illustreert dit:

Uitgediept

Stel dat de entiteit “order” door de volgende functies als volgt bewerkt wordt:
Aanmaken order (C); Annuleren order (D); Deellevering (U); Overzicht orders (R); Voorraadplanning (R)

De standaard dekking van CRUD wordt dan al bereikt met het volgende testgeval:

Aanmaken order	(C)	
Voorraadplanning	(R)	
Deellevering	(U)	
Overzicht orders	(R)	
Annuleren order	(D)	
Overzicht orders	(R)	

Hiermee zou echter de volgende fout niet gevonden worden: Na een deellevering klopt de voorraadplanning niet meer, omdat die (onterecht) de gehele order als geleverd beschouwt. Deze fout zou wel gevonden worden met de zwaardere variant, welke met het volgende testgeval bereikt wordt:

Aanmaken order	(C)	
Overzicht orders	(R)	
Voorraadplanning	(R)	
Deellevering	**(U)**	**(veroorzaakt fout in “Voorraadplanning”)**
Overzicht orders	(R)	

Voorraadplanning	**(R)**	**(fout wordt ontdekt)**
Annuleren order	(D)	
Overzicht orders	(R)	
Voorraadplanning	(R)	

14.3.8 Statistisch gebruik: Operational profiles en Load profiles

Bij deze dekkingsvorm is het niet de bedoeling om het systeemgedrag in afzonderlijke situaties te testen, maar om het realistische gebruik van het systeem *statistisch verantwoord* te simuleren. Om te kunnen testen of het systeem bestand is tegen het realistisch gebruik ervan, moet dat realistisch gebruik op een of andere wijze gespecificeerd zijn. Dergelijke specificaties worden vaak “profielen” genoemd. De twee bekendste zijn:

- Operational profiles
  Deze beschrijven welke transacties hoe vaak door een gebruiker uitgevoerd worden.
  Het testen van operational profiles heeft tot doel om te onderzoeken: “Blijft het systeem correct werken als transacties al heel *vaak en langdurig* uitgevoerd zijn?”
- Load profiles
  Deze beschrijven de belasting waaronder het systeem werkt in termen van hoeveel gebruikers er tegelijkertijd het systeem gebruiken.
  Het testen van load profiles heeft tot doel om te onderzoeken: “Blijft het systeem correct werken als *veel* transacties door *veel* gebruikers tegelijk uitgevoerd worden?”

Beide profielen worden hieronder verder toegelicht.

Operational profiles

Een operational profile beschrijft in kwantitatieve termen hoe het systeem gebruikt wordt door een bepaald type gebruiker. Dit begrip is geïntroduceerd door John Musa en voor een uitgebreide uitleg over operational profiles wordt verwezen naar zijn werk [Musa, 1998]. Hieronder volgt een korte toelichting.

Een operational profile beschrijft het realistisch gebruik door antwoord te geven op de vraag: “Als het systeem zich in *deze toestand* bevindt, hoe groot is dan de *kans* dat *deze actie* door de gebruiker uitgevoerd zal worden?” In de literatuur wordt in plaats van *toestand* en *actie* meestal gesproken over *history class* en *event*. Een operational profile geeft een statistisch gemiddelde over hoe ‘de gebruiker’ omgaat met het systeem. Als er verschillende types gebruikers te onderkennen zijn die een significant verschillend statisch gemiddeld gedrag vertonen, is het verstandig om per type gebruiker een aparte operational profile op te stellen.

Uitgediept

Zo zou bijvoorbeeld voor een systeem voor “internet bankieren” de volgende types gebruikers onderkend kunnen worden:

- scholieren
- volwassenen privé
- zakelijke gebruikers

Voor het opstellen van een operational profile worden de volgende stappen doorlopen:

1. Bepaal de lijst van mogelijke events.
   Dit omvat in principe alle acties en gebeurtenissen waarmee de systeemfunctionaliteit geactiveerd wordt. Meestal zijn dit de functies die door gebruikers of gekoppelde systemen uitgevoerd worden. Als er verschillende functies gedefinieerd zijn die veel met elkaar te maken hebben en bovendien dezelfde waarschijnlijkheid hebben om uitgevoerd te worden, is het verstandig om deze functies te groeperen. Dat houdt de operational profile overzichtelijk.
2. Bepaal de history classes van het systeem
   Een history class beschrijft de toestand waarin het systeem zich bevindt.
   Het zegt iets over wat er gebeurd is (de historie) met het systeem. Het optreden van een event (uitvoeren van een

functie) heeft meestal tot gevolg dat het systeem in een andere history class terecht komt.

Uitgediept

Bijvoorbeeld voor een DVD-recorder zijn als history classes gedefinieerd: “standby”, “opnemen”, “afspelen”, “snel vooruit”, “snel achteruit”. Als in de history class “afspelen” de gebruiker op de knop “vooruitspoelen” drukt (een event), gaat het systeem over in history class “snel vooruit”. Echter als de gebruiker deze knop indrukt in de history class “opnemen”, blijft het systeem in dezelfde history class.

In het algemeen is het zinvol om een nieuwe history class te benoemen als de kansverdeling voor het optreden van de volgende event significant verandert. Het is gebruikelijk om voor de eerste history class de initiële toestand van het systeem te kiezen.

3. Bepaal voor iedere history class de kansverdeling van de events.
   Met andere woorden: Hoe groot is de kans dat deze event uitgevoerd wordt als het systeem in deze history class zit.
   Bij iedere history class hoort de som van alle kansen op 1 (of 100%) uit te komen.

Het resultaat van deze stappen kan weergegeven worden in een matrixvorm, zoals geïllustreerd in figuur 14.10. Hierbij betekent bijvoorbeeld de term $P_{2,1}$ “de kans dat in history class 2 door de gebruiker event 1 uitgevoerd wordt”.

	Event 1	**Event 2**	**…**	**Event m**
History class 1	$P_{1,1}$	$P_{1,2}$	…	$P_{1,m}$
History class 2	$P_{2,1}$	$P_{2,2}$	…	$P_{2,m}$
…	…	…	…	…
History class n	$P_{n,1}$	$P_{n,2}$	…	$P_{n,m}$

Figuur 14.10: Algemene invulling van een operational profile.

Er zijn verschillende manieren om aan de benodigde informatie voor een operational profile te komen, bijvoorbeeld:

- Overnemen van een bestaand operational profile van een systeem met vergelijkbare functionaliteit (bijvoorbeeld het voorgaande systeem).
- Alsnog bijhouden van gegevens (logging) over hoe vaak iedere functie gebruikt wordt.
- Interviewen van gebruikers.

Let wel, het gaat hier niet om zeer nauwkeurige metingen, maar om schattingen die een voldoende realistisch beeld geven van het gemiddeld gebruik van het systeem.

Testen van operational profiles

Een testgeval bestaat in dit geval uit een keten van events die door de gebruiker uitgevoerd worden. De lengte van een testgeval is een keuze van de tester, maar is typisch in de orde van enkele tientallen events achter elkaar. Het testgeval begint in de initiële toestand (history class 1). Daarna zal het systeem in andere history classes terecht komen, afhankelijk van de events die uitgevoerd worden.

Voor het samenstellen van ieder testgeval worden de volgende stappen doorlopen:

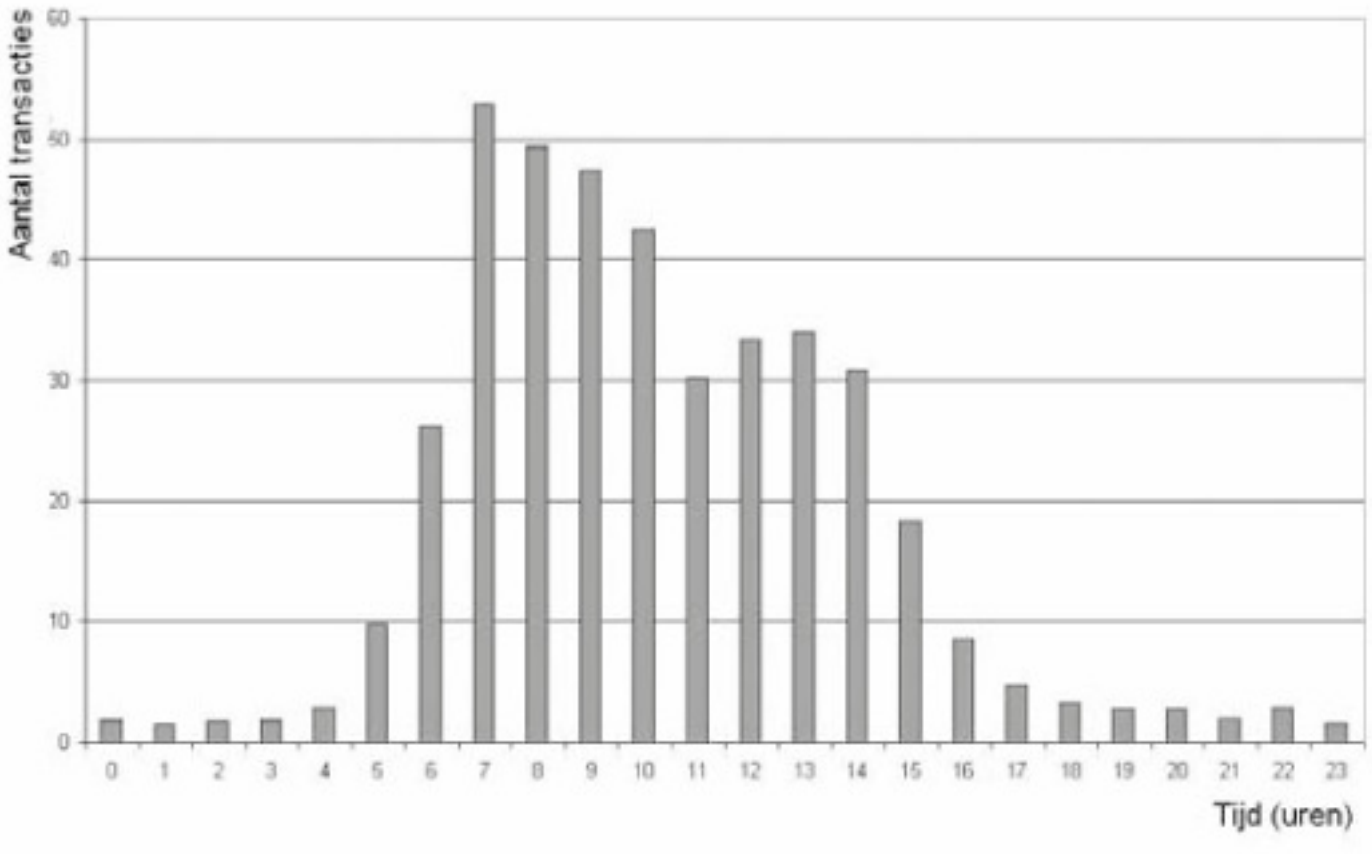
1. Begin in history class 1
2. Kies het volgende event dat uitgevoerd wordt.
   Dit wordt rechtstreeks afgeleid uit de operational profile. Met behulp van een random generator wordt een getal tussen 0 en 100% gegenereerd en de overeenkomende event uit de operational profile bepaald. Hierdoor wordt bereikt dat bij grote hoeveelheden testgevallen de events statistisch verantwoord (volgens de kansverdeling uit de operational profile) in de testset voorkomen.
3. Bepaal in welke volgende history class het systeem daardoor terecht komt.
   Dit wordt bepaald door hoe het systeem functioneel werkt. Het moet daarom afgeleid worden uit de functionele specificaties van het systeem.
4. Herhaal stappen 2 en 3 totdat de gewenste lengte van het testgeval bereikt is.

Om statistisch verantwoord het realistisch gebruik volgens de operational profile te testen, is een grote hoeveelheid

testgevallen nodig die elk een grote hoeveelheid acties (events) uitvoeren. Dit is handmatig vrijwel niet te realiseren. Het verdient aanbeveling om de benodigde testgevallen geautomatiseerd te genereren. Dit wordt hier niet verder uitgewerkt, maar meer informatie hierover is onder andere te vinden in het boek “Testing Embedded Software” [Broekman, 2003].

Load profiles

Met load profiles wordt de mate weergegeven waarmee de systeemresources (CPU, geheugen, netwerk capaciteit) in werkelijkheid belast worden. De aangebrachte belasting wordt meestal weergegeven in termen van aantal gebruikers of aantal malen dat een transactie in een bepaalde periode uitgevoerd wordt. Normaal gesproken wordt een systeem niet continu even zwaar belast, maar varieert de belasting gedurende een tijdsperiode: Er zijn piek-uren en dal-uren binnen een etmaal. Tijdens weekends is vaak een andere belasting dan op doordeweekse dagen. En in vakanties en op feestdagen kan de belasting van het systeem er nog weer anders uitzien. In figuur 14.11 is een voorbeeld weergegeven van een load profile gedurende een etmaal.



Figuur 14.11: Voorbeeld van een load profile, waarbij het aantal transacties per uur gedurende een volledig etmaal weergegeven is.

Voor het opstellen van een load profile wordt informatie uit de volgende bronnen gecombineerd:

- Met behulp van specifieke tools (monitors) de belasting van het systeem meten. De verantwoordelijkheid hiervoor ligt typisch bij een afdeling “Technisch systeembeheer”.
- Interviewen van gebruikers. Feitelijk komt het neer op de volgende vraag: “Welke transacties voer je uit? Hoe vaak en wanneer?”

Testen van load profiles

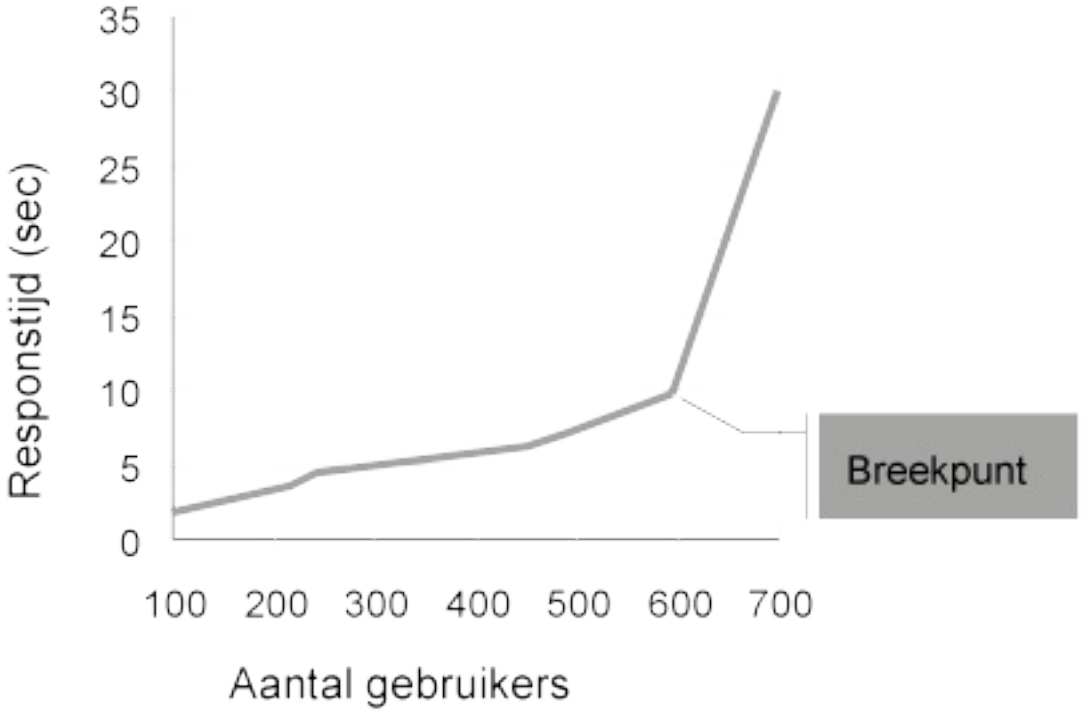
Het testen van load profiles valt in de categorie “performance testen” (zie ook paragraaf 10.3.3 “Performance”) en is een testspecialisme op zich. Hoewel het ook handmatig kan, wordt meestal gebruik gemaakt van tools die een bepaalde gedefinieerde belasting van het systeem genereren. Door de tools wordt op verschillende manieren een realistische belasting *gesimuleerd*, zoals:

- Creëren van virtuele users.
  Een virtuele user is een klein programma dat een gebruiker simuleert. Op één PC kunnen vele van dergelijke programmaatjes tegelijk draaien. Dit voorkomt dat voor iedere user daadwerkelijk een separate fysieke PC aanwezig

moet zijn. Dit wordt vooral toegepast om het totale systeem inclusief netwerk aan een bepaalde belasting te onderwerpen.
- Transacties aanbieden via de database-management interface.
  Dit creëert een zekere belasting van de back-end van het systeem zonder de front-end of het netwerk te overbelasten. Hiermee kan rechtstreeks gemeten worden of de database-server goed gedimensioneerd is.

Er zijn verschillende soorten performance tests die elk een ander doel hebben. De bekendste zijn:

- Testen bij een normaal of gemiddeld gebruik.
  Het doel is hier om te onderzoeken of de beschikbare systeemresources voldoende zijn voor de 'gebruikelijke' omstandigheden. Het idee hierbij is, dat het zakelijk voordelig kan zijn om voor de zeldzame keren dat er 'uitzonderlijke zware belasting optreedt', er tijdelijk extra resources ingezet kunnen worden.
- Testen bij piekbelasting.
  Het doel is hier om te onderzoeken of er voldoende systeemresources zijn voor zelfs de zwaarste omstandigheden die in de praktijk kunnen voorkomen.
- Meten van het breekpunt.
  Dit wordt ook wel "stress testen" genoemd. Het doel is hier om te onderzoeken wat de maximale belasting is waaronder het systeem nog acceptabel presteert. Bij een bepaalde systeemconfiguratie wordt de belasting steeds verder opgevoerd, terwijl de responstijd gemeten wordt. Dit kan in een grafiek uitgezet worden. Op het moment dat de grafiek een scherpe knik omhoog laat zien, neemt de responstijd onevenredig snel toe (de respons 'stort in') en is het breekpunt bereikt. Zie ook figuur 14.12.



Figuur 14.12: Voorbeeld van het meten van het breekpunt: de maximale belasting waarbij het systeem nog acceptabele prestaties levert.

14.3.9 Goedpaden / Foutpaden

Bij goed ontworpen systemen zijn controles ingebouwd die het systeem beschermen tegen ongeldige situaties, zoals bijvoorbeeld:

- Aanbieden van ongeldige invoer;
- Uitvoeren van een actie zonder dat aan de bijbehorende randvoorwaarden voldaan wordt.

In dergelijke situaties moet het systeem een correcte foutafhandeling uitvoeren. Dat wil zeggen dat het systeem geen onvoorspelbare of ongewenste resultaten mag produceren, maar op een gecontroleerde manier zijn verwerking moet afbreken.

Het doel van het testen van goedpaden/foutpaden is om bij iedere gedefinieerde foutsituatie zowel de geldige als de ongeldige situaties te testen. Een ongeldige situatie hoort te leiden tot een correcte foutafhandeling, terwijl een geldige

situatie door het systeem geaccepteerd dient te worden zonder foutafhandeling.

Het is niet het doel om alle mogelijke variaties binnen de geldige situaties te testen. Bijvoorbeeld bij het testen van een functie “Aanmaken order” wordt volstaan met éénmaal vast te stellen dat het systeem een toegestane order correct aanmaakt. Als men ook wil testen hoe de tientallen variaties in toegestane orders door het systeem behandeld worden, kunnen hiervoor andere dekkingsvormen toegepast worden, zoals equivalentieklassen, modified condition/decision coverage, pairwise testing, enzovoorts.

Inventariseren testsituaties: goedpaden/foutpaden

De testsituaties zijn de verzameling van alle goedpaden en foutpaden bij iedere gedefinieerde foutafhandeling. Voor het inventariseren welke foutafhandelingen getest moeten worden, kan in principe uit twee bronnen geput worden:

- Vanuit de gegevens
  “Zijn er bepaalde waarden of combinaties van waarden gedefinieerd die voor de betreffende functionaliteit niet toegestaan zijn?”
  Feitelijk is hier sprake van een speciale toepassing van het principe van *equivalentieklassen*: Ieder beschouwd gegeven wordt verdeeld in één geldige en één of meer ongeldige equivalentieklassen.
- Vanuit de processtappen
  “Zijn er in het proces of algoritme specifieke controlestappen benoemd die voorafgaan aan de verwerking?” Feitelijk is hier sprake van beslispunten met de standaarduitkomsten: Doorgaan met de processtappen (goedpad); Produceren van foutmelding en afbreken van verwerking (foutpad).

Aandachtspunten bij het dekken van de foutpaden

Bij een foutafhandeling kunnen meerdere voorwaarden een rol spelen bij het bepalen of de betreffende situatie toegestaan is of niet. Dat betekent dat er ook meerdere foutpaden bestaan, namelijk één voor elke voorwaarde waar niet aan voldaan wordt. Er is echter slechts één goedpad, namelijk als wel aan alle voorwaarden voldaan wordt.

Bij het testen van de foutpaden moet er voor gezorgd worden, dat er *niet meer dan één ongeldige situatie tegelijk* getest wordt. Als er meer ongeldige testsituaties in één testgeval gecombineerd worden, treedt het risico op van de zogenaamde “maskering” van fouten: Bij het afhandelen van de eerste ongeldige situatie breekt het systeem de verwerking reeds correct af, waardoor de tweede ongeldige situatie niet meer getest wordt. Dit wordt geïllustreerd met het volgende voorbeeld:

Uitgediept

Voor het mogen verstrekken van een lening moet aan de volgende twee voorwaarden voldaan worden:

- Leeftijd moet groter of gelijk zijn aan 18;
- Leenbedrag mag niet hoger zijn dan het jaarsalaris.

Stel dat beide foutafhandelingen met één testgeval getest zouden worden, waarbij de leeftijd kleiner is dan 18 en het leenbedrag groter dan het jaarsalaris. Het systeem produceert terecht een foutmelding nadat het heeft vastgesteld dat de leeftijd niet aan de voorwaarde voldoet. De controle op het leenbedrag wordt nu overgeslagen. Als in het systeem de foutafhandeling op leenbedrag verkeerd of zelfs helemaal niet geïmplementeerd is, zal dit met dat ene testgeval niet ontdekt worden.

Bijkomend nadeel van testgevallen met meer dan één ongeldige situatie is, dat als de foutmelding niet expliciet meldt wat er mis is, er niet aangegeven kan worden door welke ongeldige situatie de foutmelding veroorzaakt is.

14.3.10 Afvinklijst

Definitie

Bij de dekkingsvorm “afvinklijst” worden alle situaties getest die zonder structuur in een lijst opgesomd zijn.

Deze dekkingsvorm is bijvoorbeeld van toepassing op:

- Testen van requirements.
  De requirements beschrijven de eisen aan het systeem, zonder in detail te gaan over hoe dat precies gerealiseerd wordt. Bijvoorbeeld: “Betalingen moeten ook in buitenlandse valuta gedaan kunnen worden.” In principe worden voor elk requirement één of meer testgevallen gemaakt waarmee precies dat requirement getest wordt.
  Sommige requirements zijn niet testbaar en moeten dan uit de afvinklijst verwijderd worden. Bijvoorbeeld: “Met dit systeem moet het marktaandeel van het bedrijf met 20% toenemen.”
- Testen van use cases.
  Hier geldt in grote lijnen hetzelfde als bij requirements, behalve dat het hier gaat om gebruiksscenario’s in plaats van systeemeisen.
- Testen van gebruiksvriendelijkheidsaspecten.
  Voorbeelden hiervan zijn: begrijpelijke foutmeldingen; mogelijkheid van “undo” op laatste actie; maximaal 10 velden per scherm. Zie ook de tien heuristics van Nielsen in paragraaf 10.3.2 “Usability”.

Dit levert een checklist van aspecten die getest moeten worden bij iedere functie of scherm van het systeem en die tijdens het testen afgevinkt kan worden.

Het is een zeer eenvoudige dekkingsvorm waarbij de testgevallen zonder tussenstappen rechtstreeks afgeleid worden. Omdat er geen sprake is van een structuur (dus geen afhankelijkheden tussen de benoemde situaties), kan iedere regel in de afvinklijst afzonderlijk en rechtstreeks getest worden en is de volgorde van de testgevallen niet van belang. Echter het concreet uitwerken tot fysieke testgevallen is niet altijd eenvoudig en kan veel werk betekenen. Daarnaast moet men beseffen dat met de afvinklijst in het algemeen slechts een lichte dekking bereikt wordt. Bijvoorbeeld in het geval van testen van requirements geeft het slechts de zekerheid dat ieder requirement een keer getest is. Dat wil nog niet zeggen dat aangetoond is dat de requirements in alle beoogde situaties correct geïmplementeerd zijn.

Het is ook een dekkingsvorm die weinig last heeft van eventuele wijzigingen in de testbasis (en daarmee de afvinklijst zelf). Als er een regel in de afvinklijst bij komt, wordt er een testgeval toegevoegd waarmee de betreffende regel getest wordt.

Deze dekkingsvorm biedt ook de mogelijkheid om op eenvoudige wijze te sturen hoe ver het testen moet gaan, en te meten hoe ver het testen gevorderd is:

- In de plannings- of voorbereidingsfase worden de regels in de afvinklijst geprioriteerd, bijvoorbeeld met H(oog) - M(iddel) – L(aag). Een andere populaire notatiewijze voor prioriteiten is de MOSCOW notatie: Must have; Should have; Could have; Would have. In de uitvoeringsfase worden dan de regels (en dus de bijbehorende testgevallen) in volgorde van prioriteit uitgevoerd.
- Tijdens testuitvoering kan de tot dan toe behaalde dekkingsgraad gerapporteerd worden. Deze wordt als volgt gemeten: dekkingsgraad = (aantal geteste regels) / (totaal aantal regels) Dit heeft een sterke overeenkomst met voortgang (en is zelfs exact gelijk als testgevallen 1:1 zijn met de regels uit de afvinklijst):
  voortgang = (aantal uitgevoerde testgevallen) / (totaal aantal testgevallen).

14.4 Een basisset testontwerptechnieken

14.4.1 Inleiding

De plaats van testontwerptechnieken in het testproces

Definitie

Een testontwerptechniek is een gestandaardiseerde werkwijze om vanuit een bepaalde testbasis testgevallen af te leiden die een bepaalde dekking bereiken.

In de fase “Planning” wordt voor elke testvorm bepaald of en zo ja welke testontwerptechnieken toegepast gaan worden. Dit is uitgebreid beschreven in de paragrafen 6.2.5 “Bepalen teststrategie” en 6.2.8 “Toewijzen testeenheden en

testtechnieken". De keuze voor een bepaalde testontwerptechniek hangt voornamelijk af van:

- de systeemkenmerken (meestal de kwaliteitsattributen) die getest moeten worden.
- de zwaarte waarmee deze systeemkenmerken getest moeten worden.
- de beschikbare relevante informatie over deze systeemkenmerken. Dit definieert de testbasis voor de testontwerptechniek.
- de geschikte dekkingsvormen - en daarmee de basistechnieken - die op de testbasis toegepast kunnen worden.

Testontwerptechnieken worden toegepast in de fase "Specificatie", waarin de testgevallen en testscripts worden gespecificeerd in vijf generieke stappen (zie paragraaf 6.6.1 "Opstellen testspecificaties"):

1. Identificeren testsituaties
2. Opstellen logische testgevallen
3. Opstellen fysieke testgevallen
4. Vaststellen uitgangssituatie
5. Opstellen testscript

De eerste twee stappen resulteren in het logisch testontwerp waarmee de gewenste dekking gegarandeerd wordt. De laatste drie stappen resulteren in het fysiek testontwerp dat zorgt voor een gedegen voorbereiding van de testuitvoering. Zie ook paragraaf 14.2.1 "Testsituatie, testgeval en testscript".

Opzet en structuur van deze paragraaf

Deze paragraaf biedt een gevarieerde set testontwerptechnieken waarmee de meeste organisaties direct aan de slag kunnen. Deze lijst is niet uitputtend. Als aanvulling hierop kan een testorganisatie literatuur raadplegen, zoals [Beizer, 1990] en [Copeland, 2003], of zelf nieuwe technieken creëren, hetgeen aan het eind van deze inleiding beschreven wordt.

De beschrijving van iedere testontwerptechniek in deze paragraaf bestaat uit de volgende onderwerpen:

- *Omschrijving*
  Dit geeft een kernachtig beeld van de techniek door in het kort de belangrijkste kenmerken te schetsen, zoals
  – het doel van de test;
  – de benodigde testbasis;
  – de toegepaste basistechnieken;
  – hints voor variaties in de techniek.
- *Aandachtspunten bij de stappen*
  Hier wordt toegelicht hoe de generieke stappen specifiek worden ingevuld voor de betreffende testontwerptechniek. De eerste twee stappen, waarin de testsituaties en logische testgevallen ontworpen worden, krijgen de nadruk, omdat hier het grootste onderscheid bestaat tussen de verschillende testontwerptechnieken.

  De volgende twee stappen, waarin de logische testgevallen concreet uitgewerkt worden tot fysieke testgevallen, zijn voor de meeste testontwerptechnieken in het algemeen hetzelfde en krijgen daarom minder nadruk. Zij zijn reeds grotendeels beschreven in de paragrafen 6.6.1 "Opstellen testspecificaties" en 14.2.1 "Testsituatie, testgeval en testscript".

  De laatste stap "opstellen testscript" wordt in deze paragraaf niet uitgewerkt, omdat de invulling hiervan te organisatie-specifiek is en niet verschilt per testontwerptechniek. Meer hierover is reeds beschreven in eerdergenoemde paragrafen 6.6.1 en 14.2.1.

  Waar mogelijk is een concreet voorbeeld in detail uitgewerkt. Dit kan gebruikt worden als testspecificatiesjabloon voor de betreffende testontwerptechniek.

Voor gedetailleerde uitleg over de toegepaste basistechnieken dient paragraaf 14.3 "Dekkingsvormen en basistechnieken" geraadpleegd te worden.

Creëren van nieuwe testontwerptechnieken

Testontwerptechnieken komen vaak in vele varianten en combinaties voor.

Immers, een testontwerptechniek is een specifieke toepassing van de algemene basistechnieken in een bedrijfsspecifieke situatie. Afhankelijk van welke systeemkenmerken een bedrijf belangrijk vindt en op welke manier het systeemgedrag gedefinieerd is, kan een bedrijf zijn eigen testontwerptechniek uitwerken die daar het beste bij past. Bij het creëren van zo'n 'nieuwe' testontwerptechniek moeten achtereenvolgens de volgende vragen beantwoord worden:

- Wat moet er getest worden?
  Dit zegt iets over de kwaliteitsattributen of testvorm waar de testontwerptechniek voor bedoeld is.
- Wat voor informatie is hierover beschikbaar?
  Alle beschikbare relevante informatie moet gezocht en verzameld worden.
  Dit definieert de testbasis voor de testontwerptechniek.
- Wat voor soort te testen situaties moeten hoe zwaar gedekt worden?
  Dit moet uitmonden in het definiëren van één of meer dekkingsvormen en daaraan gerelateerde basistechnieken.
- In welke vorm moet het testontwerp vastgelegd worden?
  Worden de testspecificaties vastgelegd als tekst, in tabellen of spreadsheets of in een specifiek testtool?

Overzicht van de testontwerptechnieken

Tabel 14.3 vat de belangrijkste kenmerken van de hier behandelde testontwerptechnieken samen. Dit kan helpen bij het onderling vergelijken van de technieken en het selecteren van de meest geschikte.

Testontwerptechniek	Dekkingsvormen / basistechnieken	Testbasis	Kwaliteitsattribuut / Testvorm
Beslistabeltest	Multiple condition coverage	Afzonderlijke condities of beslistabellen, zonder structuur	Detail-functionaliteit
Datacombinatietest	Equivalentieklassen en (Multiple condition coverage of Pairwise testing)	Alle soorten testbasis	Overkoepelende functionaliteit Detail-functionaliteit
Elementaire Vergelijkingentest	Modified condition / decision coverage	Gestructureerde functionele specificaties, zoals pseudo-code	Detailfunctionaliteit
Error Guessing	—	Alle soorten testbasis	Diverse
Exploratory Testing	Diverse, naar keuze	Alle soorten testbasis	Diverse
Gegevenscyclustest	CRUD en Decision coverage	CRUD-matrix Data-integriteitsregels	Overkoepelende functionaliteit Connectiviteit Inpasbaarheid
Procescyclustest	Pad-dekking testmaat-2	Gestructureerde beschrijving van bedrijfs- of werkprocessen	Inpasbaarheid
Real Life Test	Statistisch verantwoorde simulatie	Operational profiles Load profiles	Bruikbaarheid Connectiviteit Performance
Semantische Test	Modified condition / decision coverage	In- en uitvoerspecificaties Business-rules	Functionaliteit / Validatietest
Syntactische Test	Afvinklijst	In- en uitvoerspecificaties Attribuut-beschrijvingen	Functionaliteit / Validatietest Gebruiksvriendelijkheid
Use cases Test	Afvinklijst	Use Case	Inpasbaarheid Bruikbaarheid Gebruiksvriendelijkheid

Tabel 14.3: Overzicht van de belangrijkste kenmerken van de beschreven testontwerptechnieken.

14.4.2 Beslistabeltest (BTT)

Omschrijving

De beslistabeltest is een grondige techniek voor het testen van detailfunctionaliteit. De benodigde testbasis bevat condities of beslistabellen. De vorm en structuur van deze testbasis is van ondergeschikt belang voor het kunnen toepassen van de beslistabeltesttechniek.

De beslistabeltest richt zich op het grondig afdekken van de condities en niet op het combineren van functionele paden. De basistechniek die hierbij wordt gebruikt, is:

- Beslispunten: multiple condition coverage.

Variaties op de beslistabeltest kunnen worden gecreëerd door het toepassen van de basistechniek:

- Beslispunten: condition coverage, decision coverage of condition/decision coverage
  Hiermee kunnen beslispunten *lichter* worden getest.
- Grenswaardenanalyse

Hiermee kunnen de mogelijkheden van een conditie *zwaarder* worden getest.

Het is een techniek die vooral zal worden gekozen voor het testen van zeer belangrijk geachte functies en/of complexe berekeningen.

Aandachtspunten bij de stappen

In deze paragraaf wordt de beslistabeltest stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 “Inleiding”) als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat bij iedere stap laat zien hoe deze techniek in zijn werk gaat.

De testbasis bestaat uit beslistabellen, pseudo-code, een procesbeschrijving of andere (functionele) beschrijvingen, waarin condities voorkomen. De condities én de resultaten worden in een beslistabel gezet. In tabel 14.4 is de algemene vorm weergegeven van een beslistabel na stap “2. Opstellen logische testgevallen”.

Identificatie tabel				
Testsituaties	**1**	**2**	**..**	**n**
Conditie 1	0	0	..	1
Conditie 2	0	~	..	0
Conditie ..	0	..	~	0
Conditie n	0	1	..	0
Resultaat 1	X	..	..	..
Resultaat ..	..	..	..	..
Resultaat n	..	..	..	..
Niet mogelijk	..	..	X	..

Tabel 14.4: Algemene vorm van een beslistabel.

Iedere kolom van de beslistabel vormt een testsituatie. Het deel boven de dubbele streep vormt de situatiebeschrijving en het deel onder de streep het gevolg c.q. de resultaten.

De condities kunnen de waarden “0” of “1” hebben (zie ook paragraaf 14.3.3).

De waarde “1” betekent dat de conditie waar is, de waarde “0” betekent dat de conditie onwaar is. Verder kan nog de waarde “~” worden toegekend. Dit betekent dat de waarde van de conditie niet van belang is. Onder de dubbele streep bevatten de cellen een “X”, of ze zijn leeg. Als er een “X” staat treedt het betreffende resultaat in die testsituatie op, als een cel leeg is treedt het betreffende resultaat niet op in die testsituatie. Er zijn meerdere resultaten per testsituatie mogelijk. Met “Niet mogelijk” wordt aangegeven dat de testsituatie niet fysiek uitvoerbaar is, bijvoorbeeld omdat bepaalde waarden van condities elkaar uitsluiten.

Voorbeeld

Bij het bestellen van koffiecapsules via het internet worden de verzendkosten berekend. Deze bestaan uit de standaard verzendkosten vermeerderd met een afstandstoeslag. Onderstaande tekst geeft bijbehorende procesbeschrijving.

Verzendkostenberekening:

Berekening standaard verzendkosten
Als er 200 of meer capsules zijn besteld en als de betalingswijze “automatische incasso” is, dan worden er geen verzendkosten berekend. Als er minder dan 200 capsules zijn besteld of als de betalingswijze niet “automatische incasso” is, dan wordt er €10 aan verzendkosten berekend.

Berekening afstandstoeslag
Als het afleveradres van de capsules binnen een straal van 50 km van Utrecht ligt, dan wordt er geen afstandstoeslag berekend. Als het afleveradres op 50 km of meer van Utrecht, maar wel in Nederland ligt, dan wordt er een afstandstoeslag van €5 berekend. Als het afleveradres buiten Nederland ligt, dan wordt er een afstandstoeslag van €15 berekend. (hoogste bedrag wordt berekend)

Hieronder is per stap uitgewerkt hoe de beslistabeltest wordt toegepast op dit proces:

1. Identificeren testsituaties

Voor het vullen van de tabel worden in stap "1-Identificeren testsituaties" de volgende activiteiten uitgevoerd:

- opsporen van condities in de testbasis;
- opstellen van een conditielijst;
- opsporen van resultaten in de testbasis en deze toevoegen aan de conditielijst;
- vullen van de beslistabel.

De activiteiten worden hieronder toegelicht:

Het opsporen van condities kost enig speurwerk. Vaak wordt in de testbasis een conditie voorafgegaan door woorden zoals "mits", "indien" en "als" en kan hierop worden gezocht.

Voorbeeld

De tester heeft in de procesbeschrijving de condities onderstreept.

Verzendkostenberekening:

Berekening standaard verzendkosten
Als er 200 of meer capsules zijn besteld én als de betalingswijze "automatische incasso" is, dan worden er geen standaard verzendkosten berekend. Als er minder dan 200 capsules zijn besteld of als de betalingswijze niet "automatische incasso" is, dan wordt er €10 aan standaard verzendkosten berekend.

Berekening afstandstoeslag
Als het afleveradres van de capsules binnen een straal van 50 km van Utrecht ligt, dan wordt er geen afstandstoeslag berekend. Als het afleveradres op 50 km of meer van Utrecht, maar wel in Nederland ligt, dan wordt er een afstandstoeslag van €5 berekend. Als het afleveradres buiten Nederland ligt, dan wordt er een afstandstoeslag van €15 berekend.

Vervolgens wordt er een conditielijst opgesteld. Als de testbasis een beslistabel is, kunnen de condities vaak één op één worden overgenomen. Bij het opstellen van de lijst worden de volgende regels toegepast. Deze regels zijn opgesteld om de beslistabellen leesbaar en overzichtelijk te houden:

- Een conditie is enkelvoudig (dus zonder "EN" of "OF" constructies).
- Een conditie wordt positief geformuleerd (om "niet niet" combinaties te vermijden).
- Probeer het aantal condities per tabel vijf of lager te houden (dit zijn maximaal al $2^5 = 32$ testkolommen). Splits bij meer condities de tabel in meerdere tabellen.

Voorbeeld

Bij de verzendkostenberekening komt de tester tot de volgende conditielijst:

Berekening standaard verzendkosten
C1 bestelling ≥ 200 capsules
C2 betalingswijze = "automatische incasso"

Berekening afstandstoeslag
C3 afstand < 50 km van Utrecht
C4 land = Nederland

Het komen tot een conditielijst vereist vaak enige interpretatie van de beschrijving. Vaak lijken er meer condities nodig te zijn om een bepaalde situatie te kunnen bereiken. Onderzoek dan of er inderdaad sprake is van een aanvullende conditie of dat een bepaalde situatie kan worden gerealiseerd door het niet waar zijn van één of meer van de al onderkende condities.

Als de conditielijst bekend is, worden de resultaten hieraan toegevoegd. Ook het opsporen van de resultaten kost enig

speurwerk. Vaak wordt in de testbasis een resultaat voorafgegaan door woorden zoals “dan” en “anders”.

Voorbeeld

De tester heeft in de procesbeschrijving de resultaten onderstreept.

Verzendkostenberekening:

Berekening standaard verzendkosten
Als er 200 of meer capsules zijn besteld én als de betalingswijze “automatische incasso” is, dan worden er geen standaard verzendkosten berekend. Als er minder dan 200 capsules zijn besteld of als de betalingswijze niet “automatische incasso” is, dan wordt er €10 aan standaard verzendkosten berekend.

Berekening afstandstoeslag
Als het afleveradres van de capsules binnen een straal van 50 km van Utrecht ligt, dan wordt er geen afstandstoeslag berekend. Als het afleveradres op 50 km of meer van Utrecht, maar wel in Nederland ligt, dan wordt er een afstandstoeslag van €5 berekend. Als het afleveradres buiten Nederland ligt, dan wordt er een afstandstoeslag van €15 berekend.

De tester voegt de resultaten toe aan de conditielijst:

Berekening standaard verzendkosten
C1 Bestelling ≥ 200 capsules
C2 Betalingswijze = “automatische incasso”
R1 Standaard verzendkosten := 0
R2 Standaard verzendkosten := 10

Berekening afstandstoeslag
C3 Afstand < 50 km van Utrecht
C4 Land = Nederland
R3 Afstandstoeslag := 0
R4 Afstandstoeslag := 5
R5 Afstandstoeslag := 15

Nu zowel condities al resultaten bekend zijn, wordt de beslistabel volgens de multiple condition coverage gevuld.

Voorbeeld uitwerking

Aangezien het totaal aantal condities vier bedraagt, heeft de tester besloten deze in één tabel op te nemen[1]. De conditielijst en het vullen van de tabellen volgens de multiple condition coverage levert de tabel met testsituaties voor het verzendkostenvoorbeeld op zoals weergegeven in tabel 14.5:

Verzendkostenberekening (testsituaties)																
St.verz.kosten / afst.toeslag	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
C1 Bestelling ≥ 200 capsules	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
C2 Betalingswijze = “aut. inc.”	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
C3 Afstand < 50 km van Utrecht	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
C4 Land = Nederland	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
R1 St. verzendkosten := 0									X	X	X	X				
R2 St. verzendkosten := 10	X	X	X	X	X	X	X	X					X	X	X	X
R3 Afstandstoeslag :=0			X	X	X	X					X	X	X	X		
R4 Afstandstoeslag :=5		X					X			X					X	
R5 Afstandstoeslag :=15	X			X	X			X	X			X	X			X

Tabel 14.5: Tabel met testsituaties, gevuld volgens multiple condition coverage.

Het lezen van een tabel wordt vaak lastig gevonden. Testsituatie 7 bijvoorbeeld, wordt als volgt gelezen:
Besteller heeft minder dan 200 capsules besteld EN heeft betalingswijze “automatische incasso” gekozen EN afleveradres ligt op 50 km of meer van Utrecht EN afleveradres ligt in Nederland. De verzendkosten bedragen €10 standaard verzendkosten plus €5 afstandstoeslag is €15.

Het met enen en nullen vullen van een tabel kan op vele manieren. De wijze van vullen zoals gebruikt in bovenstaande tabel “Verzendkostenberekening” maakt het fysiek maken eenvoudiger (zie uitgediept hieronder voor toelichting).

Uitgediept

Let op de manier van ‘handig’ vullen van de beslistabel “Verzendkostenberekening”.

Hierdoor verandert er per kolom slechts één conditie van waarde (in de literatuur bekend onder de naam: “Gray-code”). Dit is prettig bij het fysiek maken van de testgevallen: copy and paste en één waarde veranderen. Voor de handige vulling beginnen we in de onderste conditierij met één 0, vervolgens achtereenvolgens twee keer een 1, twee keer een 0 net zolang tot aan de laatste, die krijgt waarde 0. In de één na onderste rij beginnen we met twee keer 0, vervolgens achtereenvolgens vier keer 1, vier keer 0 net zolang tot aan de laatste twee, die krijgen waarde 0. Zo gaan we door met de hele tabel, bij iedere rij zijn de nul- en één-stukken twee keer zo lang als bij de vorige.

Uitgediept

In plaats van het vullen van de beslistabel volgens de basistechniek multiple condition coverage, kan tijdens de strategiebepaling voor een (lichtere) variant zijn gekozen. Deze techniek wordt op zowel de condities als de resultaten toegepast. Als voorbeeld zijn in tabel 14.6 de testsituaties uitgewerkt voor “verzendkostenberekening” volgens de condition/decision coverage:

Verzendkostenberekening (testsituaties)			
St.verz.kosten / afst.toeslag	**1**	**2**	**11**
C1 Bestelling ≥ 200 capsules	0	0	1
C2 Betalingswijze = “aut. inc.”	0	0	1
C3 Afstand < 50 km van Utrecht	0	0	1
C4 Land = Nederland	0	1	1
R1 St. verzendkosten := 0			X
R2 St. Verzendkosten := 10	X	X	
R3 Afstandstoeslag := 0			X
R4 Afstandstoeslag := 5		X	
R5 Afstandstoeslag := 15	X		

Tabel 14.6: Tabel met testsituaties, gevuld volgens condition/decision coverage.

Met de drie kolommen worden alle mogelijke uitkomsten van elke conditie én van elk resultaat minimaal één keer gedekt. De kolommen 1 en 11 dekken alle condities af en kolom 2 is nodig om ook resultaat 4 af te dekken.

Er zijn meer combinaties van kolommen mogelijk die aan de condition/decision coverage voldoen (bijvoorbeeld: 3, 9, 10 en 2, 11, 16).

Uitgediept

Naast condities die slechts de waarden waar of onwaar kunnen bezitten, bestaan er ook parameters met meer dan 2 mogelijke waarden (beter: equivalentieklassen). Er zijn twee manieren om daar mee om te gaan:

1. Splits de parameters op in meerdere condities, die alleen waar of onwaar kunnen zijn. Vervolgens kan de in deze paragraaf beschreven aanpak worden gebruikt.
2. Voeg net zoveel kolommen toe als er mogelijke equivalentieklassen zijn, waarbij de inhoud van de overige conditierijen niet wijzigt. Stel dat in het voorbeeld uit drie betalingswijzen kon worden gekozen: automatische incasso, acceptgiro en contant. Dan zou als voorbeeld de ‘oude’ testsituatie 1 bij deze aanpak tot de volgende drie ‘nieuwe’ testsituaties leiden:

St.verz.kosten / afst.toeslag	**1**	**2**	**3**
C1 Bestelling ≥ 200 capsules	0	0	0
C2 Betalingswijze	“aut. inc.”	“acc. giro”	“contant”
C3 Afstand < 50 km van Utrecht	0	0	0
C4 Land = Nederland	0	0	0
R2 St. Verzendkosten := 10	X	X	X

R5 Afstandstoeslag := 15	X	X	X

Voor meer informatie hierover wordt verwezen naar “N-wise testing” in paragraaf 14.3.5 “Orthogonale arrays en Pairwise testing” en naar theorieboeken zoals [Copeland, 2003].

2. Opstellen logische testgevallen

De testsituaties (kolommen) uit stap 1 zijn direct de logische testgevallen.

Een logisch testgeval mag echter geen ‘elkaar uitsluitende condities’ bevatten, want dat maakt het testgeval in zichzelf inconsistent en daarmee onuitvoerbaar. In de stap van testsituaties naar logische testgevallen moeten eventueel onuitvoerbare testgevallen worden opgespoord. Deze testgevallen krijgen in de tabel het resultaat “Niet mogelijk”.

Voorbeeld

In het verzendkostenvoorbeeld sluiten de condities C3 en *niet*-C4 elkaar uit. Immers, er liggen binnen een straal van 50 km van Utrecht geen buitenlandse plaatsen. Hierdoor zijn 4 van de 16 logische testgevallen niet uitvoerbaar. De logische testgevallen zijn uitgewerkt in tabel 14.7.

Verzendkostenberekening (logische testgevallen)																
St.verz.kosten / afst.toeslag	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
C1 Bestelling ≥ 200 capsules	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
C2 Betalingswijze = “aut. inc.”	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
C3 Afstand < 50 km van Utrecht	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
C4 Land = Nederland	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
R1 St. verzendkosten := 0									X	X	X					
R2 St. Verzendkosten := 10	X	X	X			X	X	X						X	X	X
R3 Afstandstoeslag := 0			X			X					X			X		
R4 Afstandstoeslag := 5		X					X			X					X	
R5 Afstandstoeslag := 15	X							X	X							X
Niet mogelijk				X	X							X	X			

Tabel 14.7: Tabel met logische testgevallen.

Ga zelf na dat de logische testgevallen 4, 5, 12 en 13 niet uitvoerbaar zijn.

Voor de logische testgevallen met als resultaat “Niet mogelijk” bestaat geen fysiek testgeval. Het verdient de voorkeur deze kolommen niet te verwijderen, zodat er geen misverstand kan ontstaan waarom voor een bepaald logisch testgeval geen fysiek testgeval aanwezig is.

3. Opstellen fysieke testgevallen

Bij een fysiek testgeval krijgen alle gegevens die een rol spelen in de condities een concrete invulling. Hiervoor kan de tabel met de logische testgevallen simpel worden aangepast voor het maken van de fysieke testgevallen. In de tabel met fysieke testgevallen beschrijft elke genummerde kolom een fysiek testgeval en bevat(ten) de laatste rij(en) de resultaatvoorspelling(en). Voor het invullen worden:

- de “0” of de “1” uit de tabel vervangen door een fysieke waarde;
- op de plaats van een “X” de fysieke waarde ingevuld.

Aandachtspunt bij de eerste bullet is dat in de tabel van fysieke testgevallen geen condities meer, maar gegevens staan. Een bepaald gegeven kan in meer condities voorkomen. In logische testgevallen komen die dan in meerdere rijen voor, terwijl dat gegeven in de tabel voor fysieke testgevallen natuurlijk maar 1 keer voorkomt. Hiernaast is het mogelijk dat er aanvullende uitwerkingen noodzakelijk zijn. Bijvoorbeeld door het geven van een concrete invulling aan een afgeleid gegeven (zie onderstaand voorbeeld).

Voorbeeld uitwerking

Voor het fysiek maken is het van “Afstand van Utrecht” afgeleide gegeven “Afleverplaats” nodig. Het resultaat is

weergegeven in tabel 14.8, waarbij als voorbeeld 8 van de 12 logische testgevallen fysiek zijn uitgewerkt.

Verzendkostenberekening (fysieke testgevallen)								
	1	**2**	**6**	**7**	**9**	**10**	**11**	**16**
Aantal capsules	199	199	199	199	200	200	200	200
Betalingswijze	contant	contant	aut.inc.	aut.inc.	aut.inc.	aut.inc.	aut.inc.	contant
Afstand van Utrecht	178	182	10	182	178	182	10	178
Afleverplaats	Brussel	Heerlen	Zeist	Heerlen	Brussel	Heerlen	Zeist	Brussel
Land	Bel	Ned	Ned	Ned	Bel	Ned	Ned	Bel
St. verzendkosten	10	10	10	10	0	0	0	10
Afst. toeslag	15	5	0	5	15	5	0	15
Totale verzendkosten	25	15	10	15	15	5	0	25

Tabel 14.8: Tabel met fysieke testgevallen.

De logische testgevallen 4, 5, 12 en 13 zijn niet uitvoerbaar en kunnen dus ook niet fysiek worden gemaakt.

Uitgediept

De beslistabeltest kan worden verzwaard door het toepassen van de grenswaardenanalyse. De keuze hiervoor wordt bij het opstellen van de logische testgevallen als extra voorwaarde opgenomen. In het voorbeeld zou deze voorwaarde opgenomen kunnen worden bij het aantal capsules en de afstand. De eis is dan dat bij aantal capsules in ieder geval de waarden 199, 200 en 201 (bijvoorbeeld in de kolommen 9, 10 en 11) en bij afstand in ieder geval de waarden 49, 50 en 51 (bijvoorbeeld in de kolommen 1, 2 en 7) in de fysieke testgevallen voorkomen. Het aantal testgevallen verandert bij deze aanpak niet.

Een andere mogelijkheid is om voor elke waarde een aparte kolom op te nemen. Dit is een grondige methode die alle combinaties test, maar wel arbeidsintensief is. Het aantal testgevallen neemt bij deze aanpak toe.

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

Voorbeeld

De gegevens van de besteller die relevant zijn om een bestelling te kunnen plaatsen, moeten in het informatiesysteem aanwezig zijn.

14.4.3 Datacombinatietest (DCT)

Omschrijving

De datacombinatietest (DCT) is een veelzijdige techniek voor het testen van de functionaliteit op zowel detailniveau als op overkoepelend systeemniveau. In de embedded wereld staat deze techniek ook wel bekend als de “Classification Tree Method” die ontwikkeld is door Grochtmann en Grimm en onder andere beschreven is in “Testing Embedded Software” [Broekman, 2003].

Voor de DCT is geen specifieke testbasis nodig. Iedere vorm van informatie over de functionaliteit van het systeem is bruikbaar:

- Formele systeemdocumentatie, zoals functioneel ontwerp, logisch gegevensmodel en requirements;
- Informele documentatie, zoals handleidingen, folders, vooronderzoeken en memo’s;
- Materiedeskundigheid die niet gedocumenteerd is, maar ‘in de hoofden van experts’ zit.

Het feit dat materiedeskundigheid bruikbaar is als testbasis, maakt deze techniek ook geschikt voor situaties waarin specificaties niet compleet of niet actueel zijn of zelfs in het geheel niet beschikbaar.

Wegens het sterk informele karakter van deze techniek, wordt de kwaliteit van de hiermee ontworpen testgevallen voor een groot deel bepaald door de expertise en vakbekwaamheid van de betrokkenen. Daarom wordt de DCT bij voorkeur uitgevoerd door een team van 2 à 5 personen met een *mix van expertises*: test-, materie- en systeem-deskundigheid.

Tip

Organiseer een 'creatieve sessie', zoals een brainsorm-sessie of metaplan-sessie, waarbij de tester als moderator van het proces optreedt. Nodig voor deze sessie één expert uit van iedere relevante discipline, bijvoorbeeld een gebruiker, een beheerder en een systeemontwikkelaar of -architect. De experts leveren de inhoudelijke informatie welke door de tester wordt gestructureerd en omgezet in testsituaties en testgevallen.

Bij de DCT worden de testsituaties bepaald door vanuit de gegevens te redeneren welke variaties getest moeten worden. De hierbij gebruikte basistechniek is

- Equivalentieklassen.

Afhankelijk van de afgesproken zwaarte van de test kan de dekking uitgebreid worden door de equivalentieklassen van twee of meer verschillende gegevens volledig met elkaar te combineren. Daarbij kunnen de volgende basistechnieken gebruikt worden:

- Pairwise testing;
- Volledige beslistabel (multiple condition coverage) op geselecteerde gegevens.

Daarnaast kan gekozen worden voor een verzwaring van de test door het toepassen van *grenswaardenanalyse*. Dit kan ook selectief toegepast worden, door voor specifieke gegevens de grenswaarden als aparte equivalentieklasse te definiëren.

Door zijn veelzijdigheid kan de datacombinatietest worden gekozen voor het testen van zowel zeer belangrijk geachte functies als van systeemdelen die 'slechts vluchtig getest moeten worden'.

Aandachtspunten bij de stappen

In deze paragraaf wordt de datacombinatietest stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 "Inleiding") als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat bij iedere stap laat zien hoe deze techniek in zijn werk gaat.

Voorbeeld

Dit voorbeeld gaat over een functie waarmee plaatsen voor een vliegreis gereserveerd kunnen worden: Bij deze functie voert de gebruiker een aantal gegevens in over de samenstelling van de groep (volwassenen, kinderen, baby's) en de reis (zoals datum, bestemming). Daarna kan de gebruiker kiezen volgens welke criteria er gezocht moet worden naar een passende vlucht. Het systeem toont dan de lijst met mogelijke vluchten of geeft een melding als er geen enkele vlucht bestaat die aan de criteria voldoet en nog over de benodigde plaatsen beschikt.

Deze functie moet met gemiddelde zwaarte getest worden met behulp van de DCT.

1. Identificeren testsituaties

Het identificeren van testsituaties is de creatieve stap in het proces en wordt bij voorkeur uitgevoerd door een team waarin verschillende expertises vertegenwoordigd zijn. In deze stap worden de volgende activiteiten uitgevoerd:

- Bepaal de gegevens die van invloed zijn op de functionaliteit
  Dit zijn niet automatisch alle gegevens die door de functie *gebruikt* worden.
  Het gaat om de gegevens die van invloed zijn op *variaties* in het systeemgedrag. Dit zijn dan ook de gegevens waarvoor equivalentieklassen bepaald kunnen worden.
  De gedefinieerde gegevens kunnen bestaan uit entiteiten, attributen of functionele begrippen in algemene zin.

- Bepaal voor ieder gegeven de equivalentieklassen
  Zie hiervoor paragraaf 14.3.4.
- Bepaal de relaties tussen de gegevens
  Sommige gegevens zijn pas van invloed op het systeemgedrag onder een bepaalde voorwaarde, namelijk als een ander gegeven een waarde uit een specifieke equivalentieklasse heeft. Dat betekent dat de mogelijke variaties van dat eerstgenoemde gegeven gecombineerd *moeten* worden met die specifieke waarde van dat laatstgenoemde gegeven. In de voorbeeld-uitwerking is een dergelijke relatie zichtabaar tussen de gegevens “zoekcriterium” en “vliegt op die bestemming”.

Het resultaat kan grafisch weergegeven worden in een zogenaamde classificatieboom:

- Gegevens die logisch bij elkaar horen, kunnen gegroepeerd worden onder een overkoepelend begrip, zoals bijvoorbeeld “persoonsgegevens” of “werkgevers-types”.
- Onder ieder gegeven worden de equivalentieklassen gehangen, als takken aan de boom.
- Relaties tussen de gegevens kunnen eenvoudig weergegeven worden, door de betreffende delen van de classificatieboom rechtstreeks onder de betreffende equivalentieklasse te hangen.

Het maken van de classificatieboom waarmee de testsituaties geïdentificeerd worden, is een iteratief en interactief proces waarbij de betrokkenen elkaar inspireren, corrigeren en aanvullen. Hoe ver dit proces gaat, is de vrijheid en de verantwoordelijkheid van het team. Een testmanager die dit goed onder controle wil houden, zal zorgen voor een concrete opdrachtomschrijving voor het team en regelmatige terugkoppeling verlangen over de resultaten. Eventueel wordt gedefinieerd welke gegevens in aanmerking komen om ‘volledig gecombineerd getest’ te worden. Dat wil zeggen dat alle mogelijke combinaties van alle equivalentieklassen van die gegevens getest moeten worden. Hoeveel van dergelijke combinaties gedefinieerd mogen worden, is afhankelijk van de afgesproken zwaarte van de test.

Tip

Het volgende kan als richtlijn gebruikt worden:

Licht	Geen of slechts één gegevenspaar.
Gemiddeld	Twee of meer gegevensparen. Dit biedt een schaal van toenemende zwaarte die eindigt bij “pairwise testing”.
Zwaar	Gemiddelde zwaarte + grenswaardenanalyse.

In plaats van combinaties van twee gegevens (gegevenspaar) kan ook gedefinieerd worden dat alle combinaties van drie gegevens (gegevens-triplet) getest moeten worden. Dit moet dan behandeld worden als een verhoging van de zwaartecategorie.

Voorbeeld uitwerking

Voor de functie “plaatsreservering” is door het team de volgende classificatieboom uitgewerkt (zie figuur 14.13):

Figuur 14.13: Voorbeeld van een classificatieboom.

Bij de uitwerking zijn onder andere de volgende aspecten van belang:

- Een passagier kan volwassene, kind of baby zijn. Een baby heeft geen eigen zitplaats in het vliegtuig.
- Een geplande reis die langer dan een jaar is, zou tot verwarring kunnen leiden over de datum voor de terugreis.
- Een vliegmaatschappij kan al of niet op de bedoelde bestemming vliegen. Dit is uitsluitend relevant onder de voorwaarde dat als zoekcriterium deze maatschappij opgegeven is.

Er zijn twee gegevensparen gedefinieerd, passend bij de afgesproken gemiddelde zwaarte, die volledig gecombineerd getest moeten worden:

- kind – baby (4 verplichte combinaties)
- bestemming – vliegt op die bestemming (6 verplichte combinaties)

2. Opstellen logische testgevallen

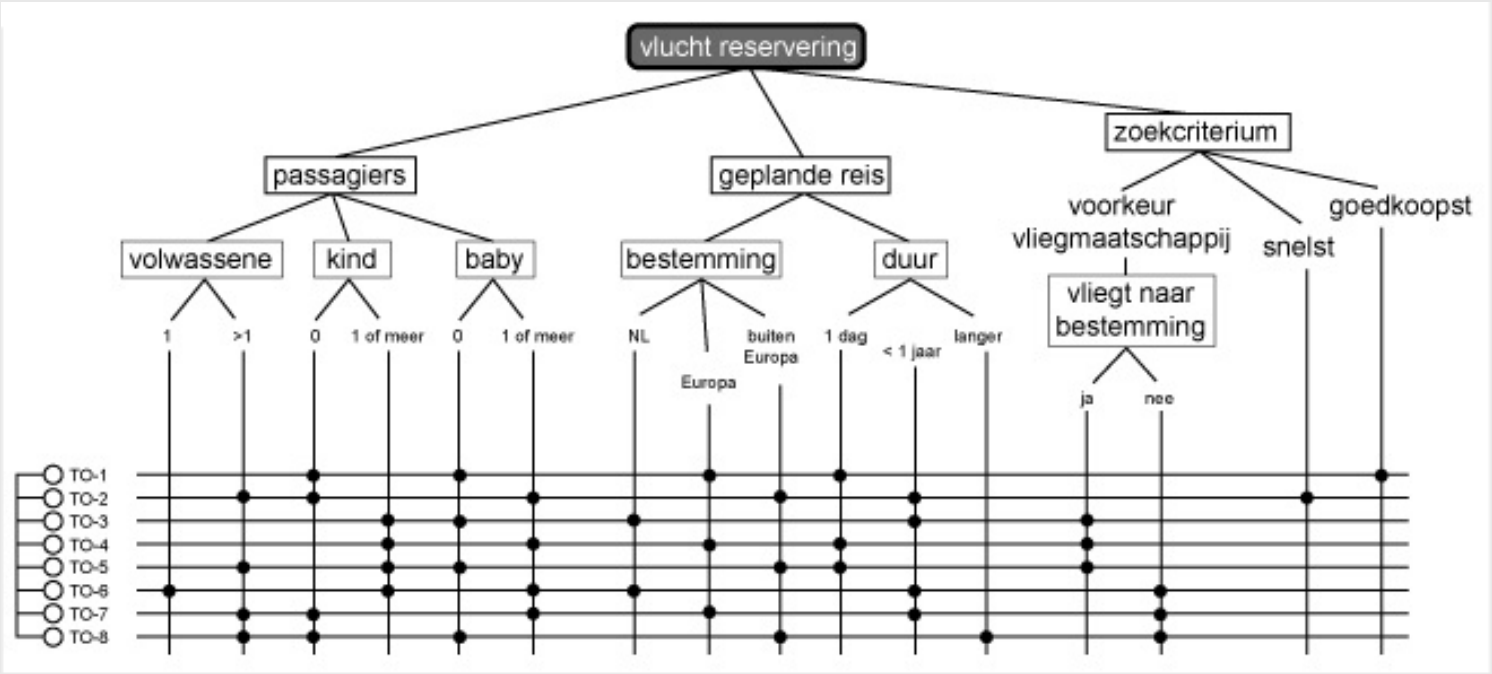
Bij een logisch testgeval wordt voor *ieder* gegeven uit de classificatieboom precies *één* van de equivalentieklassen gedekt. Gezamenlijk moeten de logische testgevallen in ieder geval alle equivalentieklassen van alle gegevens afdekken. Afhankelijk van de gekozen zwaarte moeten zij eventueel ook alle combinaties van equivalentieklassen van bepaalde gegevens afdekken. In principe zijn er twee manieren om dit overzichtelijk weer te geven:

- In tabelvorm.
  Deze manier wordt meestal gebruikt als gekozen is voor “pairwise testing” (zie paragraaf 14.3.5). Tools voor pairwise testing leveren hun resultaat meestal direct in tabelvorm.
- Grafische weergave van een zogenaamde ‘classification tree’.

Deze is met name bruikbaar als er gekozen is voor de lichtste vorm (zonder combinaties) of voor het selectief toepassen van de “volledige beslistabel” dekking. Bij voorkeur wordt hierbij gebruik gemaakt van een grafisch tool. Via internet is het freeware tool “Classification Tree Editor” te downloaden (www.systematic-testing.com), die speciaal voor dit doel ontwikkeld is door DaimlerChrysler.

Voorbeeld uitwerking

De logische testgevallen zijn weergegeven met behulp van de classificatieboom (zie figuur 14.14):



Figuur 14.14: Voorbeeld van een classificatieboom met logische testgevallen.

Hierbij is rekening gehouden met de twee gegevensparen die volledig gecombineerd getest moeten worden. Testgevallen TG-1 t/m TG-4 dekken het gegevenspaar “kind - baby” af, terwijl het gegevenspaar “bestemming – vliegt op bestemming” afgedekt wordt door de testgevallen TG-3 t/m TG-8.

N.B. Voor het behalen van de minimale dekking, waarbij uitsluitend alle equivalentieklassen gedekt worden zonder combinaties van gegevensparen, zijn 4 logische testgevallen hier voldoende, bijvoorbeeld TG-1, TG-2, TG-3 en TG-8.

3. Opstellen fysieke testgevallen

Bij het fysiek maken van de testgevallen moeten concrete waarden gekozen worden voor alle invoergegevens. Deze invoergegevens komen niet altijd exact overeen met de begrippen die in de classificatieboom gehanteerd zijn. Zo kan in de classificatieboom het begrip “periode” benoemd zijn, terwijl de te testen functie de gegevens “startdatum” en “einddatum” verwacht.

Bij ieder fysiek testgeval hoort ook een concrete resultaatvoorspelling. Deze hangt in het algemeen echter ook af van de overige gegevens en systeeminstellingen die bij de gekozen uitgangssituatie horen.

Voorbeeld uitwerking

Om het principe te illustreren zijn in tabel 14.9 vier testgevallen fysiek uitgewerkt. Bij ieder testgeval zijn de fysieke waarden voor alle benodigde invoergegevens gedefinieerd en de resultaatvoorspelling concreet beschreven.

	TG-1	TG-2	TG-3	TG-6
klantnaam	Jansen	Breugel	Voort	Hansma
#volwassenen	1	2	3	1
#kinderen	0	0	1	4
#baby’s	0	2	0	1
bestemming	Frankrijk-CdG	Singapore	Nederland-Eindhoven Airport	Nederland-Eindhoven Airport
datum vertrek	12-02-2006	14-02-2006	15-02-2006	16-02-2006
datum retour	12-02-2006	15-02-2006	15-04-2006	23-02-2006
zoekcriterium	goedkoopst	snelst	KLM	Senegal Airlines
Resultaatvoorspelling				Melding: “maatschappij vliegt niet op keuzebestemming”
maatschappij	Korean Air	Canada Air	KLM	
vluchtnummer	KA0455	CA0833	KL1288	
prijs	€ 44	€ 865	€ 83	

Tabel 14.9: Fysieke testgevallen voor “plaatsreservering”.

Voor het voorspellen van de resultaten van ieder fysiek testgeval, is het nodig om precies te weten welke vluchten en prijzen in de database opgeslagen zijn. Deze stap gaat dus hand-in-hand met de volgende stap “vaststellen uitgangssituatie”.

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

Voorbeeld uitwerking

De volgende database moet geladen worden:“TST_RES_03”. Deze bevat in het bijzonder de situatie dat maatschappij “Senegal Airlines” wel bestaat, maar niet “Eindhoven Airport” als bestemming kent.
Zet de systeemdatum op 01-02-2006.

14.4.4 Elementaire Vergelijkingentest (EVT)

Omschrijving

De elementaire vergelijkingentest (EVT) is een grondige techniek voor het gedetailleerd testen van de functionaliteit. De benodigde testbasis is pseudocode of een vergelijkbare specificatie waarin de beslispunten en functionele paden gedetailleerd en gestructureerd uitgewerkt zijn.

De EVT richt zich op het grondig afdekken van de beslispunten en niet op het combineren van functionele paden. De basistechniek die hierbij gebruikt wordt is

- Beslispunten: modified condition/decision coverage

Variaties op de EVT kunnen gecreëerd worden door het toepassen van de volgende basistechnieken:

- Beslispunten: multiple condition coverage
  Hiermee kunnen de mogelijkheden binnen de (eventueel specifiek geselecteerde) beslispunten zwaarder getest worden.
- Grenswaardenanalyse
  Hiermee kunnen de mogelijkheden binnen de (eventueel specifiek geselecteerde) beslispunten zwaarder getest worden.
- Pairwise testing
  Hiermee wordt het testen van mogelijke combinaties van functionele paden toegevoegd.

Het is een techniek die vooral zal worden gekozen voor het testen van zeer belangrijk geachte functies en/of complexe berekeningen.

Aandachtspunten bij de stappen

In deze paragraaf wordt de elementaire vergelijkingentest stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 “Inleiding”) als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat bij iedere stap laat zien hoe deze techniek in zijn werk gaat.

Voorbeeld uitwerking

In dit voorbeeld gaat het om een functie (taak) waarbij de gegevens van de autobezitter op een scherm worden ingevoerd en vervolgens op aanvraag een berekening wordt gemaakt van de premie die de autobezitter moet betalen voor zijn autoschadeverzekering. Afhankelijk van een aantal variabelen wordt de hoogte van de premie vastgesteld. De onderstaande pseudo-code geeft een gedetailleerde functionele beschrijving hiervan:

```
ALS                 leeftijd < 18 jaar OF rijbewijs ingevorderd
DAN                 foutboodschap
ANDERS    ALS       leeftijd < 25 jaar EN aantal jaren rijbewijs < 3
          DAN       premie := 1.500
          ANDERS    premie := 800
          EINDALS
          ALS       autoleeftijd < 2 OF (autoleeftijd ≥ 5
                    EN schade in afgelopen 3 jaar ≥ 2.500)
                    OF leeftijd ≥ 70
          DAN       verhoog premie met 500
          EINDALS
EINDALS
```

Hieronder is per stap uitgewerkt hoe de elementaire vergelijkingentest wordt toegepast op deze functie:

1. Identificeren testsituaties

De testbasis bestaat uit pseudo-code of een vergelijkbare formele functiebeschrijving, zodat deze rechtstreeks overgenomen kan worden in deze stap. Zo niet, dan moet een extra activiteit uitgevoerd worden om de bestaande specificaties om te zetten in pseudo-code.

De beslispunten in de pseudo-code worden van een unieke identificatie voorzien. Het is gebruikelijk hiervoor de codes B1, B2, … te gebruiken (of B01, B02,… als er sprake is van veel beslispunten).

Voorbeeld uitwerking

Er zijn drie beslispunten, die hieronder gestructureerd weergegeven zijn:

```
B1  ALS                 leeftijd < 18 jaar OF rijbewijs ingevorderd
    DAN                 foutboodschap
B2  ANDERS  ALS         leeftijd < 25 jaar EN aantal jaren rijbewijs < 3
            DAN         premie := 1.500
            ANDERS      premie := 800
            EINDALS
B3          ALS         autoleeftijd < 2 OF (autoleeftijd ≥ 5
                        EN schade in afgelopen 3 jaar ≥ 2.500)
                        OF leeftijd ≥ 70
            DAN         verhoog premie met 500
            EINDALS
    EINDALS
```

Per beslispunt wordt de basistechniek voor MCDC (modified condition/ decision coverage) toegepast. De hieruit resulterende testsituaties worden genummerd. De combinatie van dit nummer en het beslispunt geeft een unieke identificatie van de testsituaties (zoals: B1-1, B1-2, enzovoorts). Bij het nummeren wordt begonnen met de testsituaties uit de kolom “1” (True) en daarna die uit de kolom “0” (False).

Voor ieder beslispunt worden de testsituaties in een aparte tabel in detail uitgewerkt. De rijen van de tabel bevatten de gegevens of parameters die in de condities van het beslispunt voorkomen. Een kolom geeft dan aan welke eisen aan iedere parameter gesteld zijn voor de betreffende testsituatie.

Voorbeeld uitwerking

B1 **A OF B**	**1** **foutboodschap**	**0**
A: leeftijd < 18	1 (1)	0 0 (3)
B: rijbewijs ingevorderd	0 1 (2)	~~0 0~~

B2 **A EN B**	**1** **Premie = 1.500**	**0** **Premie = 800**
A: leeftijd < 25	1 1 (1)	0 1 (2)
B: # jaren rijbewijs< 3	~~1 1~~	1 0 (3)

B3 **A OF (B EN C) OF D**	**1** **Premie + 500**	**0**
A: autoleeftijd < 2	1 **0 1** 0 (1) (of 1 **1 0** 0, maar dat geeft logische tegenspraak, of 1 **0 0** 0)	0 0 1 0 (4)
B: autoleeftijd ≥ 5	0 1 1 0 (2)	~~0 0 1 0~~
C: 3-jaar-schade ≥ 2.500	~~0 1 1 0~~	0 1 0 0 (5)
D: leeftijd ≥ 70	0 **1 0** 1 (3) (of 0 **0 1** 1, of 0 **0 0** 1)	~~0 1 0 0~~

Let op bij B3! De combinatie “A = waar en B = waar” levert een logische tegenspraak en mag dus niet voorkomen in de testsituaties: Autoleeftijd moet tegelijkertijd kleiner dan 2 en groter of gelijk aan 5 zijn. Deze tegenspraak zou overigens vanzelf boven water komen bij het fysiek maken van de testgevallen.

Detailuitwerking van de afgeleide testsituaties:

Testsituaties B1	**B1-1**	**B1-2**	**B1-3**
leeftijd	< 18	≥ 18	≥ 18
rijbewijs ingevorderd	N	Y	N

Testsituaties B2	**B2-1**	**B2-2**	**B2-3**
leeftijd	< 25	≥ 25	< 25
# jaren rijbewijs	< 3	< 3	≥ 3

Testsituaties B3	**B3-1**	**B3-2**	**B3-3**	**B3-4**	**B3-5**
autoleeftijd	< 2	≥ 2	≥ 2	≥ 2	≥ 2
autoleeftijd	< 5	≥ 5	≥ 5	< 5	≥ 5
3-jaar-schade	≥ 2,500	≥ 2,500	< 2,500	≥ 2,500	< 2,500

leeftijd	< 70	< 70	≥ 70	< 70	< 70

Let op! De parameter “leeftijd” komt voor in de beslispunten B1, B2 en B3. Dit leidt tot de volgende elkaar uitsluitende testsituaties: B2-1 met B3-3; B2-3 met B3-3.

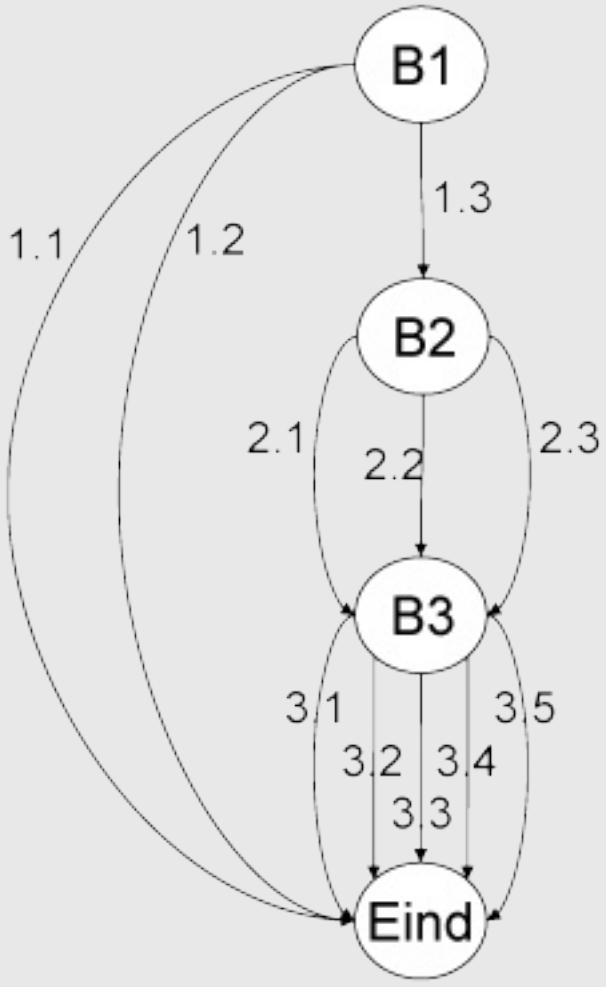
Grafische voorstelling van testsituaties

Voor sommige testers wordt het bedenken van logische testgevallen eenvoudiger met behulp van een grafische voorstelling van de testsituaties – kortweg: *Graaf*.

Daarbij wordt ieder beslispunt en eindpunt weergegeven door een cirkel en iedere testsituatie door een lijn die van de ene cirkel naar een andere gaat. Een logisch testgeval doorloopt de graaf van begin tot eind, via een aaneenschakeling van testsituaties. Tevens geeft de graaf inzicht in hoeveel testgevallen er minimaal nodig zijn om alle testsituaties af te dekken. Dit wordt namelijk bepaald door het maximaal aantal parallelle lijnen in de graaf.

Voorbeeld uitwerking

De testsituaties zijn in een graaf weergegeven in figuur 14.15.



Figuur 14.15: Graaf met testsituaties bij een EVT.

2. Opstellen logische testgevallen

Een testgeval doorloopt de functionaliteit van start tot eind en zal daarbij één of meer beslispunten op zijn pad tegenkomen. Bij ieder beslispunt zal het testgeval één van de gedefinieerde testsituaties testen.

De logische testgevallen worden samengesteld met behulp van een matrix.

De rijen bevatten de testsituaties en de kolommen bevatten de logische testgevallen. Bij ieder testgeval is met één of meer kruisjes aangegeven welke testsituaties door dit testgeval getest dienen te worden. Deze matrix dient tegelijk als controle

op de volledige afdekking van testsituaties.

Om rekening te houden met de nesting van beslispunten, zijn de kolommen “Waarde” en “Volgende” toegevoegd. Deze geven voor iedere testsituatie aan wat de uitkomst van de beslissing is (kan rechtstreeks uit de tabellen uit stap2 gehaald worden) en naar welk volgende beslispunt (of einde proces) dit leidt. Dit helpt te voorkomen dat de tester een kruisje zet bij een testsituatie waar het testgeval helemaal niet komt.

Voorbeeld uitwerking

Tabel 14.10 beschrijft de testgevallen TG-1 t/m TG-7 die de gewenste dekking geven:

Testsituatie	Waarde	Volgende	TG1	TG2	TG3	TG4	TG5	TG6	TG7
B1-1	1	Einde	X						
B1-2	1	Einde		X					
B1-3	0	B2			X	X	X	X	X
B2-1	1	B3				X			X
B2-2	0	B3			X		X		
B2-3	0	B3						X	
B3-1	1	Einde			X				
B3-2	1	Einde				X			
B3-3	1	Einde					X		
B3-4	0	Einde						X	
B3-5	0	Einde							X

Tabel 14.10: Logische testgevallen bij een EVT.

Elkaar uitsluitende testsituaties

Iedere testsituatie stelt bepaalde eisen aan één of meer parameters. Als een parameter in meerdere beslispunten voorkomt, kan het gebeuren dat een testsituatie uit het ene beslispunt eisen aan die parameter stelt die strijdig zijn met de eisen van een testsituatie uit een ander beslispunt. Bijvoorbeeld: testsituatie B2.1 eist “leeftijd < 25” en testsituatie B3.3 eist “leeftijd ≥ 70”. Deze testsituaties sluiten elkaar uit.

Een logisch testgeval mag geen “elkaar uitsluitende testsituaties” bevatten, want dat maakt het testgeval inconsistent en daarmee onuitvoerbaar. Zo’n testgeval wordt overigens automatisch ontdekt, zodra het testgeval fysiek gemaakt moet worden (zie volgende stap). Het probleem kan vervolgens eenvoudig opgelost worden, door één van de “uitsluitende testsituaties” te vervangen door een niet conflicterende testsituatie. In dit verband kan het voordelig zijn om ieder logisch testgeval eerst uit te werken tot fysiek testgeval om mogelijke “elkaar uitsluitende testsituaties” te ontdekken, voordat aan een volgend logisch testgeval begonnen wordt.

Om de kans op het ontstaan van testgevallen met elkaar uitsluitende testsituaties te reduceren, kan als optionele stap vooraf een extra analyse uitgevoerd worden:

- Inventariseer welke parameters in meerdere beslispunten voorkomen, en (per parameter) welke beslispunten dat zijn.
- Maak een opsomming van de combinaties van elkaar uitsluitende testsituaties.

3. Opstellen fysieke testgevallen

Bij een fysiek testgeval moeten alle parameters (gegevens) een concrete invulling krijgen, zodanig dat de betreffende testsituaties hiermee afgedekt worden.

Fysieke testgevallen kunnen handig beschreven worden met behulp van een matrix die als volgt opgebouwd is:

- Iedere kolom beschrijft een fysiek testgeval.
- De eerste rij geeft per testgeval aan welke testsituaties afgedekt dienen te worden.
- Daarna is er een rij voor iedere parameter waaruit het testgeval bestaat.
- Tenslotte worden één of meer rijen toegevoegd waarmee de resultaatvoorspelling concreet beschreven wordt.

Voorbeeld uitwerking

In tabel 14.11 zijn de logische testgevallen fysiek uitgewerkt, inclusief de bijbehorende resultaatvoorspelling:

Testgeval	TG1	TG2	TG3	TG4	TG5	TG6	TG7
Bevat testsituaties	B1-1	B1-2	B1-3 B2-2 B3-1	B1-3 B2-1 B3-2	B1-3 B2-2 B3-3	B1-3 B2-3 B3-4	B1-3 B2-1 B3-5
Leeftijd	**16**	**33**	**35**	**19**	**72**	**24**	**20**
Rijbewijs ingevorderd	N	J	N	N	N	N	N
# jaren rijbewijs			2	0	1	5	2
Autoleeftijd			1	12	6	3	20
3-jaar-schade			6000	4300	50	2700	2200
Resultaat:							
Foutmelding	X	X					
Premie			1300	2000	1300	800	1500

Tabel 14.11: Fysieke testgevallen bij een EVT.

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

Voorbeeld uitwerking

In de database moeten de premiebedragen zijn vastgelegd.

De gegevens van de autobezitter worden online ingevoerd en de premie wordt vervolgens direct berekend.

14.4.5 Error guessing (EG)

Omschrijving

Error guessing is het gissen naar fouten. De waarde ervan ligt in het onverwachte: er worden tests uitgevoerd die anders niet aan bod komen.

Op basis van de ervaring van de tester gaat deze op zoek naar foutgevoelige plekken in het systeem en bedenkt hiervoor passende testgevallen.

Ervaring is hierbij een breed begrip: dit kan de professionele tester zijn die fouten ‘ruikt’ in bepaalde complexe schermafhandeling, maar het is ook de gebruiker of beheerder die de uitzonderingssituaties vanuit zijn praktijk kent en wil testen of het nieuwe of aangepaste systeem hier goed mee omgaat.

Samen met Exploratory testing (paragraaf 14.4.6) is error guessing een beetje een vreemde eend in de bijt van de testontwerptechnieken. Beide technieken zijn niet gebaseerd op één van de beschreven basistechnieken en geven daarmee ook geen bepaalde dekking.

De zeer informele techniek laat de tester vrij om de testgevallen van te voren te ontwerpen of ter plekke te bedenken tijdens de testuitvoering. Vastleggen van de testgevallen is optioneel. Een aandachtspunt wanneer de testgevallen niet worden vastgelegd is de reproduceerbaarheid van de test. Vaak weet de tester niet goed meer onder welke omstandigheden een bevinding zich voordeed. Een mogelijke maatregel hiervoor is het maken van notities (een ‘testlogboek’) tijdens de test. Vanzelfsprekend worden met de test gedane bevindingen wel vastgelegd. In die gevallen moet veel aandacht worden besteed aan de omstandigheden die tot de bevinding hebben geleid, zodat de bevinding reproduceerbaar is.

Tip

Een hulpmiddel voor het kunnen reproduceren van een bevinding is het aanzetten van logging tijdens de test, waarmee de acties van de tester worden vastgelegd. Een tool voor geautomatiseerde testuitvoering kan hiervoor

gebruikt worden.

De grote vrijheid van de techniek maakt het toepassingsgebied erg groot. Error guessing kan voor het testen van elk kwaliteitsattribuut en bij elke vorm van testbasis toegepast worden.

Tip

Error guessing kan ook worden gebruikt om bijvoorbeeld gebruikers of beheerders vertrouwen te laten krijgen (of herkrijgen) in een systeem met als idee ‘als deze situaties goed verwerkt worden dan zal de rest het ook wel goed doen’.

Uitgediept

Error guessing wordt nogal eens verward met exploratory testing (paragraaf 14.4.6).

Tabel 14.12 vat de verschillen samen:

Error guessing	Exploratory testing
maakt geen gebruik van de basistechnieken	maakt afhankelijk van de situatie gebruik van de meest passende basistechniek
geschikt voor testers, gebruikers, beheerders, etc.	geschikt voor ervaren testers met kennis van de basistechnieken
de testgevallen worden in de fase Specificatie of tijdens testuitvoering ontworpen	de testgevallen worden tijdens testuitvoering ontworpen
richt zich op de uitzonderingen en moeilijke situaties	richt zich op het totaal van een te testen aspect (scherm, functie)
niet systematisch, geen enkele zekerheid over dekking	enigszins systematisch

Tabel 14.12: Verschillen tussen error guessing en exploratory testing.

Uitgediept

In de praktijk wordt voor de toegepaste testtechniek vaak error guessing gebruikt bij gebrek aan betere benaming, ‘het is geen bekende testontwerptechniek, dus is het error guessing’. Met name het testen van bedrijfsprocessen door gebruikers of het testen van requirements wordt error guessing genoemd. Hierbij wordt echter de basistechniek “afvinklijst” gebruikt, terwijl er bij error guessing geen specifieke basistechniek wordt gebruikt.

Het feit dat tests uitgevoerd worden die anders niet aan bod komen, maakt error guessing tot een waardevolle aanvulling op de overige testontwerptechnieken. Omdat error guessing geen enkele dekking garandeert, is het echter geen vervanging!

Error guessing vindt bij voorkeur later in het totale testproces plaats, wanneer de meeste normale en eenvoudige fouten er al uit gehaald zijn met de reguliere technieken. Hierdoor kan error guessing zich richten op het testen van de echte uitzonderingen en moeilijke situaties. Vanuit de teststrategie moet de testmanager normaal gesproken enige sturing geven aan het doel van error guessing, zodat doublures met andere tests voorkomen worden. Ook stelt de testmanager een bepaalde hoeveelheid tijd (timebox) en middelen beschikbaar.

Aandachtspunten bij de stappen

De stappen kunnen zowel doorlopen worden in de fase Specificatie als tijdens testuitvoering. De tester legt normaal gesproken de resultaten van de stappen niet vast, maar als grote waarde wordt gehecht aan bewijsmateriaal of overdraagbaarheid en herbruikbaarheid van de test dient dit wel te gebeuren.

1. Identificeren testsituaties

Voorafgaand aan de testuitvoering identificeert de tester de zwakke plekken waar de test zich op moet richten. Dit zijn vaak fouten in het denkproces van anderen en zaken die vergeten worden. Deze aspecten vormen de basis voor de uit te voeren testgevallen. Voorbeelden hiervan zijn:

- uitzonderingssituaties: zelden voorkomende situaties in de systeembediening, schermafhandeling of de (bedrijfs)procesgang;
- foutafhandeling: een foutsituatie forceren tijdens de afhandeling van een andere foutsituatie, onderbreking van een proces op een onverwacht moment, enzovoort;
- niet toegestane invoer: negatieve getallen, nullen, te grote waarden, te lange namen, lege (verplichte) velden, enzovoort (alleen zinvol is als er geen syntactische test op dit onderdeel uitgevoerd wordt);
- specifieke combinaties, bijvoorbeeld op het gebied van:
  - gegevens: een nog niet eerder uitgeprobeerde combinatie van invoerwaarden;
  - volgorde van transacties: bijvoorbeeld een aantal keren achter elkaar “wijzigen – annuleren wijziging – opnieuw wijzigen – annuleren – enzovoorts”
- het claimen van teveel systeemresources (geheugen, diskruimte, netwerk);
- complexe onderdelen van het systeem;
- vaak gewijzigde onderdelen van het systeem;
- onderdelen (processen/functies) van het systeem die in het verleden vaak fouten bevatten.

2. Opstellen logische testgevallen

Deze stap vindt normaal gesproken alleen bij complexere testgevallen plaats.

De tester kan dan overwegen een logisch testgeval op te stellen dat de te testen situatie afdekt.

3. Opstellen fysieke testgevallen

Deze stap vindt normaal gesproken alleen bij complexere testgevallen plaats.

De tester kan dan overwegen een fysiek testgeval op te stellen voor het logische testgeval.

4. Vaststellen uitgangssituatie

Tijdens deze activiteit kan ook vast komen te staan dat het noodzakelijk is een bepaalde uitgangssituatie op te bouwen ten behoeve van de test.

14.4.6 Exploratory testing (ET)

Omschrijving

Exploratory testing is een aantal jaren geleden door James Bach als begrip en aanpak neergezet en beschreven. Wat is exploratory testing?

Definitie

Definitie volgens James Bach [Bach, 2002]

Exploratory testing is het simultaan leren, ontwerpen en uitvoeren van tests, met andere woorden elke vorm van testen waarbij de tester zijn testen ontwerpt tijdens de testuitvoering en de informatie die wordt verkregen tijdens het testen wordt gebruikt om nieuwe en betere testgevallen te ontwerpen.

Dit betekent dat de tester steeds een stukje van het te testen systeem verkent (exploreert), nadenkt over wat getest moet of kan worden (testontwerp) en dit vervolgens ook doet (testuitvoering). Hierbij doet de tester gelijk al weer nieuwe kennis op over het systeem, denkt na over wat vervolgens getest moet worden, voert dit uit, enzovoort. Het ontwerpen en vervolgens uitvoeren van de tests gebeurt daarmee zeer dicht op elkaar. Vastleggen van het testgeval is niet nodig. Bij exploratory testing vindt dus wel degelijk testontwerp plaats in tegenstelling tot ad-hoc of ongestructureerd testen. De tester maakt afhankelijk van de te testen kenmerken en de daarover beschikbare informatie gebruik van de meest toepasselijke basistechniek. Dit stelt hoge eisen aan de tester, omdat deze een grote verzameling basistechnieken moet kunnen toepassen zonder elk techniekstapje expliciet uit te werken.

Uitgediept

Wanneer een tester een testscript uitvoert, stuit hij nogal eens op ‘verdacht’ systeemgedrag, dat wil zeggen dat het systeem er anders uitziet of anders reageert dan verwacht. Dit hoeft niet per se opgeschreven te zijn in het testscript, de verwachting hoeft zelfs niet gestoeld te zijn op bepaalde systeemdocumenten. Wanneer de tester dit verdachte gedrag nader onderzoekt, is deze explorerend aan het testen. De meeste testers zullen hierbij echter roepen dat dit vanzelfsprekend is en niet het gevoel hebben dat ze exploratory testing toepassen. Daarmee is het dus niet zozeer de vraag of iemand explorerend test, maar meer in welke mate hij dit doet. Dit maakt het lastig om te onderscheiden wat nu wel of niet exploratory testing te noemen.

Evenals error guessing (paragraaf 14.4.5) is exploratory testing wat moeilijk op één lijn te plaatsen met de overige testontwerptechnieken. Het is niet gebaseerd op één van de beschreven basistechnieken, het laat de keuze van de toe te passen basistechnieken vrij en geeft geen gegarandeerde dekking. Er is zelfs discussie te voeren óf het eigenlijk wel een testontwerptechniek is. Voornamelijk om praktische redenen is in dit boek de keuze gemaakt exploratory testing hier wel bij in te delen.

Uitgediept

In de praktijk wordt exploratory testing nog wel eens verward met error guessing. In paragraaf 14.4.5 staan de verschillen tussen beide uitgewerkt.

Exploratory testing wordt nogal eens geassocieerd met testen zonder formele testbasis als een functioneel ontwerp. Dit hoeft echter niet zo te zijn. Het is heel goed mogelijk om het toe te passen bij een goed beschreven systeem. Wel is het zo dat de techniek zich goed leent om voor gebruik wanneer er geen beschreven testbasis is. Exploratory testing legt minder nadruk op een beschreven testbasis en meer op andere manieren om te beoordelen of het testobject voldoet, onder andere door het systeem te leren kennen tijdens de testuitvoering.

Tip

Aan het gebruik van andere vormen van testbasis dan de formele systeemdocumentatie kleeft een risico. Dit is dat de tester zozeer gaat vertrouwen op deze andere in formatiebronnen dat het testen tegen de systeemdocumentatie naar de achtergrond verdwijnt. Een ongewenst eindresultaat kan dan zijn dat systeem en documentatie niet synchroon lopen. Indien gewerkt wordt met contracten kan dit contractuele problemen geven. Wanneer het systeem correct is en de documentatie niet, kan dit een onderhouds- of beheerprobleem geven. Om die reden moet bij een bevinding altijd de relatie tussen software en formele testbasis worden beschouwd.

Omgekeerd is het mogelijk dat in de systeemdocumentatie complexe, onverwachte functionaliteit beschreven is en dat die functionaliteit incorrect in het systeem is geïmplementeerd. Bij het testen op basis van andere bronnen dan de systeemdocumentatie komen deze bevindingen met betrekking tot de incorrecte implementatie er niet uit.

Een ander ongewenst resultaat kan zijn dat de testers bij gebrek aan duidelijkheid over de scope een eindeloze stroom wijzigingsverzoeken genereren onder het mom van bevindingen.

Beide zijn aandachtspunten voor de testmanager.

Wanneer wel of niet toepassen

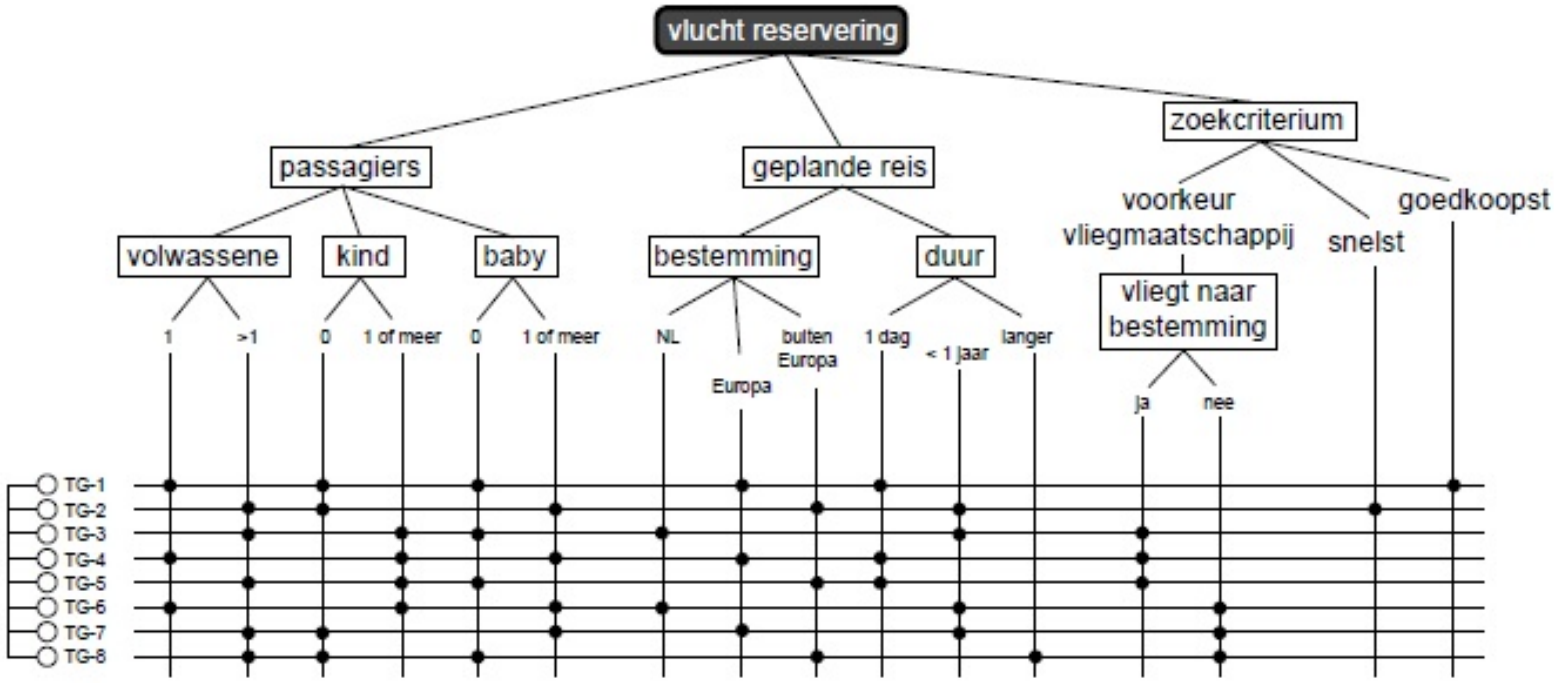
De grote vrijheid van de techniek maakt het toepassingsgebied erg groot. Het kan voor het testen van elk kwaliteitsattribuut en bij elke vorm van testbasis toegepast worden. Wel zijn er diverse omstandigheden waarin exploratory testing beter wel of niet kan worden toegepast. Onderstaand wordt dit aangegeven.

Wel toepassen

- *Wanneer ervaren testers met materiekennis beschikbaar zijn waarin men voldoende vertrouwen heeft.*
  Als testers niet hoeven te documenteren wat ze doen, is dat goedkoper dan wanneer ze wel moeten documenteren.

Keerzijden van de keuze zijn een grotere kans op onvoldoende kwaliteit van de test, langere doorlooptijd van testen op het kritieke pad van het project en minder overdraagbaarheid en herbruikbaarheid van de tests (waardoor mogelijk hogere testkosten op lange termijn). De randvoorwaarden en keerzijden worden in de overige punten van deze paragraaf toegelicht.

- *Wanneer zo goedkoop mogelijk testen verreweg de belangrijkste overweging is.*
  Omdat dit meestal het geval is, zou exploratory testing vrijwel altijd toepasbaar zijn. Deze stelling gaat echter niet op. De techniek kan ingezet worden vanuit kostenoogpunt, maar er zijn enkele belangrijke randvoorwaarden: men dient goede en ervaren testers te hebben én ook volledig vertrouwen in hen te hebben en niet de dekking en diepgang van testen aangetoond te willen zien. Keerzijden van de keuze zijn een groter risico op onvoldoende kwaliteit van de test, langere doorlooptijd van testen op het kritieke pad van het project en minder overdraagbaarheid en herbruikbaarheid van de tests (waardoor mogelijk hogere testkosten op lange termijn). De randvoorwaarden en keerzijden worden in de overige punten van deze paragraaf toegelicht.
- *Wanneer er onvoldoende gedocumenteerde testbasis is.*
  Door explorerend bezig te zijn, krijgt de tester al doende een beeld van het testobject. Omdat nadruk wordt gelegd op de inventiviteit en intuïtie van de tester, is het meer geschikt om te gebruiken wanneer er weinig systeemdocumentatie is op het moment dat testen begint of wanneer de documentatie sterk afwijkt (lees: verouderd is) van de gewenste werking van het systeem. Dit maakt de techniek ook heel geschikt bij ‘documentatielichte’ en agile methoden als Extreme programming, DSDM en (in mindere mate) bij Rational Unified Process. Aandachtspunten hierbij zijn dat het ontbreken van systeemdocumentatie een risico is dat men met het inzetten van exploratory testing niet kan opheffen en dat de testers naast testkennis dan ook moeten beschikken over veel systeem- en materiekennis omdat een referentiekader in de vorm van systeemdocumentatie ontbreekt.
- *Als aanvulling op testen volgens formelere technieken, om creatief testen te stimuleren.*
  Vaak zitten de fouten in de software op onverwachte plaatsen én zitten de fouten dicht bij elkaar (clustering). Formele testontwerptechnieken zijn gericht op het vinden van bepaalde typen fouten. Het gebruik van deze technieken kan gezien worden als een filter waar de software doorheen gaat. Met elk filter, lees: testontwerptechniek, worden bepaalde typen fouten gevonden. De vraag blijft dan of er nog veel fouten van andere typen achterblijven en hoe ernstig deze zijn. Exploratory testing kan hier aanvullend inzicht in geven. Door de informele aard is deze minder gericht op bepaalde typen fouten. Bij een fout die met exploratory testing is gevonden, moet de tester zich daarom goed afvragen of de fout uniek en op zichzelf staand is of dat de fout ook op allerlei andere plaatsen kan voorkomen of een cluster aangeeft. Hierdoor is de techniek een goede aanvulling op de formelere technieken. Figuur 14.16 maakt dit duidelijk.



Figuur 14.16: Exploratory testing als aanvulling op testen met formelere technieken.

- *Als er geen tijd beschikbaar is om de tests voor te bereiden.*
  Hoewel geen ideale situatie en één waarbij vanuit testen zeker de risico’s aangegeven moeten worden, komt dit toch regelmatig voor. Exploratory testing is dan een middel om in de korte resterende tijd het maximale aan testen te bereiken en op korte termijn globaal inzicht in de productkwaliteit te verkrijgen. Stel, de tester heeft 8 uur om een bepaalde groep van functies of schermen te testen. Wat levert dan meer op: 8 uur lang de functies op allerlei explorerende manieren doorlopen, met gebruikmaking van checklists, en in te zoomen wanneer iets er verdacht uitziet of de helft van de tijd besteden om een aantal testgevallen op herbruikbare en controleerbare manier te specificeren en

die in de resterende tijd nauwgezet te gaan uitvoeren?

Andere omstandigheden wanneer de techniek goed toegepast kan worden, zijn wanneer de testers snel willen leren hoe het systeem werkt, om de kwaliteit van andermans testen met een korte test te beoordelen, om een snelle eerste indruk te krijgen van de kwaliteit van het systeem of om een specifieke bevinding of mogelijke foutbron te onderzoeken.

Dit wil niet zeggen dat wanneer aan één van bovenstaande situaties wordt voldaan, exploratory testing gelijk dé oplossing is. Er zijn namelijk ook diverse situaties wanneer het toepassen minder geschikt is.

Niet toepassen

- Wanneer hogere eisen aan de aantoonbaarheid/verslaglegging van testen worden gesteld, bijvoorbeeld door opgelegde standaards.
  Het testproces is moeilijker beheersbaar, meetbaar en toetsbaar (auditable) omdat geen testgevallen worden gedefinieerd en aan de logging van tests lage eisen worden gesteld. Het is niet bekend wat de tester gaat doen en hoe hij het gaat doen. Achteraf is nauwelijks controleerbaar wat er gedaan is.
- Bij kritische functionaliteit waarvan het falen grote schade kan veroorzaken.
  Omdat weinig wordt vastgelegd, leunt de techniek erg op het vertrouwen in de individuele tester. De potentiële schade wanneer dit vertrouwen ongegrond blijkt te zijn, kan zo groot zijn voor bepaalde systemen of delen van een systeem, dat de organisatie dit risico niet kan of wil nemen.
- Voor onervaren testers, omdat deze de kennis en ervaring missen om zonder expliciete ondersteuning door een techniek goede testgevallen te bedenken.
- Wanneer testgevallen moeten kunnen worden uitgevoerd door een andere tester dan de bedenker of door een testtool. Of wanneer de testgevallen herbruikbaar moeten zijn, bijvoorbeeld in later onderhoud.
- Wanneer er geen rechtstreekse feedback van testuitvoering is, zodat de testresultaten niet direct beschikbaar zijn, bijvoorbeeld bij nachtelijke testruns van batchprogrammatuur.
- Bij tests die veel voorbereiding vergen zoals het testen van ingewikkelde berekeningen, performancetesten, het testen van beveiliging of testen van usability.
  Deze voorbereidingen, die zowel kunnen slaan op testgevallen, uitgangssituaties, referentietabellen als testomgevingen, kunnen beter ruim vóór testuitvoering plaatsvinden om niet onnodig veel tijd kwijt te zijn tíjdens testuitvoering.
- Wanneer het testen zo kort mogelijk op het kritieke pad van het project moet zitten.
  De tester begint bij exploratory testing laat, pas nadat het testobject is opgeleverd en doet zowel ontwerp als uitvoering van tests tijdens de fase Uitvoering. Meestal is dit op het kritieke pad van het project. Dit vergt meer doorlooptijd dan wanneer de tests al vóór oplevering van het testobject ontworpen zijn in de vorm van testscripts of checklists.
  Van belang hierbij is dan wel dat de testscripts voorbereid kúnnen worden, dus dat er een voldoende gedocumenteerde testbasis is. Overigens moet gezegd worden dat het onderhouden van testscripts tijdens de testuitvoering meestal ook extra tijd kost, waardoor het tijdsvoordeel van testscript voor het kritieke pad enigszins vermindert.

Of de techniek zinvol toegepast kan worden hangt dus af van diverse factoren. Het is aan de testmanager om dit te beoordelen. Gezien de hoge eisen aan de tester betekent dit dat exploratory testing met name ingezet wordt in testteams waarin professionele testers participeren. In de praktijk zijn dit de systeem- en acceptatietests.

Aandachtspunten bij de stappen

De generieke stappen (zie 14.4.1 “Inleiding”) zijn in principe ook van toepassing voor exploratory testing. De concrete invulling van de stappen is echter afhankelijk van welke basistechnieken door de tester tijdens testuitvoering gekozen en toegepast worden. Daarom worden de stappen in deze paragraaf niet verder uitgewerkt.

Daarnaast zijn de generieke stappen normaal gesproken alleen impliciet van toepassing en worden niet gedocumenteerd. Wanneer zwaardere eisen aan bewijsmateriaal of overdraagbaarheid worden gesteld, kan afgesproken worden dat de tester wél de testgevallen expliciet vastlegt. Dit gebeurt dan simultaan met de testuitvoering. Ook kan gekozen worden voor de variant van session based test management.

Uitgediept

Session based test management

Als groot nadeel van exploratory testing wordt vaak de onbeheersbaarheid genoemd. Om dit te ondervangen, heeft

Jonathan Bach session based test management als aanpak geïntroduceerd [Bach, 2000]. Bij op sessies gebaseerd testen wordt het te testen (onderdeel van het) systeem verdeeld in een aantal testcharters. Een testcharter kan van alles zijn, denk bijvoorbeeld aan een functie, scherm, menu, gebruikershandleiding, gebruikersvriendelijkheid of performance, of heel algemeen een gebied met mogelijke instabiliteit zoals geheugengebruik. Hiermee is een testcharter dus iets anders dan een testgeval of een testscript. Bij het testen van een testcharter worden vaak diverse testgevallen uitgevoerd. Criteria die aan een testcharter gesteld worden, zijn:

- Het heeft een te behalen testdoel
- Het stelt een eenheid van werk voor, die ongeveer tussen een half uur en vier uur in beslag neemt.
- Het is onafhankelijk testbaar, ofwel een tester kan met elke testcharter beginnen of eindigen.

De testcharters worden getest in zogenaamde testsessies. Een sessie is een tijdsperiode, normaal gesproken ook tussen een half en vier uur, waarin de tester ononderbroken één of meer testcharters kan testen. Tijdens testen legt de tester zijn/haar acties (op hoofdlijnen) vast in de vorm van notities op het sessieblad. Hierdoor zijn de tests tot op zekere hoogte herbruikbaar, omdat de hertest van een testcharter kan plaatsvinden op basis van de aantekeningen van de vorige sessie(s). Het is dan de keuze van de tester om de vorige sessie zoveel mogelijk te herhalen óf juist andere variaties te proberen.

In tegenstelling tot de testcharters die meerdere keren getest kunnen worden, zijn de testsessies eenmalig: je begint en eindigt een sessie op een bepaald moment. Wanneer je op een later moment een testcharter opnieuw moet testen, vindt dit plaats in een nieuwe sessie. Het voordeel van sessies is dat deze in de tijd begrensd zijn en daardoor beter beheersbaar. Na afloop van de sessie neemt de testmanager de sessie door met de tester in een zogenaamde debriefing om de prioriteit van de gevonden bevindingen te bepalen, leerpunten te delen en een inschatting van (overblijvende) risico's te maken.

De tester kan tijdens de sessie nieuwe testcharters bedenken. Deze worden dan toegevoegd aan de lijst met testcharters. Door sessies en testcharters te administreren, kan de voortgang van het testproces bewaakt worden en kan de buitenwereld inzicht krijgen in wat er in het testen is gebeurd en nog moet gebeuren.

14.4.7 Gegevenscyclustest (GCT)

Omschrijving

De gegevenscyclustest (GCT) is een techniek om te testen of de gegevens op consistente wijze gebruikt en bewerkt worden door verschillende functies vanuit verschillende deelsystemen of zelfs verschillende systemen. De techniek is bij uitstek geschikt voor het testen van overkoepelende functionaliteit, inpasbaarheid en connectiviteit.

Het primaire doel van de gegevenscyclustest is niet om functionele fouten in afzonderlijke functies op te sporen, maar om integratiefouten te vinden. De test richt zich op de koppeling tussen verschillende functies en de wijze waarop zij met gemeenschappelijke gegevens omgaan. De GCT is het meest effectief als de functionaliteit van de afzonderlijke functies reeds voldoende getest is. Dat is ook een belangrijke reden waarom deze test meestal in de latere fasen van acceptatietesttrajecten toegepast wordt.

De belangrijkste testbasis is de CRUD-matrix (zie paragraaf 14.3.7) en een beschrijving van de geldende integriteitsregels. Deze laatste beschrijven de randvoorwaarden waaronder bepaalde bewerkingen wel of niet zijn toegestaan, zoals bijvoorbeeld "gegeven-X mag pas gewijzigd worden als eerst het daaraan gekoppelde gegeven-Y verwijderd is". Daarnaast zijn functionele specificaties of gedetailleerde materiedeskundigheid nodig om voor ieder testgeval het resultaat te kunnen voorspellen.

De gehanteerde basistechnieken zijn

- CRUD, voor het afdekken van de levensloop van de gegevens;
- decision coverage, voor het afdekken van de integriteitsregels.

Verzwaring van de test kan bereikt worden door het toepassen van bijvoorbeeld

- zwaardere variant van de CRUD;
- modified condition/decision coverage of multiple condition coverage op de integriteitsregels.

Aandachtspunten bij de stappen

In deze paragraaf wordt de gegevenscyclustest stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 “Inleiding”) als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat tot en met het ontwerpen van de logische testgevallen laat zien hoe deze techniek in zijn werk gaat.

1. Identificeren testsituaties

De testsituaties ontstaan uit het afdekken van de CRUD en van de integriteitsregels. Beide zullen hier verder worden toegelicht.

Testsituaties met betrekking tot CRUD
De volgende activiteiten dienen uitgevoerd te worden:

- Bepaal de gegevens waarvan de levensloop getest wordt.
  Normaal gesproken betreft het hier alle gegevens die door het systeem of subsysteem gebruikt (gemaakt, gewijzigd, gelezen of verwijderd) worden. Als het aantal gegevens te groot is, kan gekozen worden voor een samenhangende subset van de gegevens.
- Bepaal de functies die gebruik maken van deze gegevens.
  Ook hier moet bepaald worden wat de scope van de test is: alle functies van het te testen systeem; een samenhangende subset hiervan; ook functies uit andere systemen die gekoppeld zijn aan het te testen systeem.
- Vul de CRUD-matrix (zie paragraaf 14.3.7)
  Als de CRUD-matrix als testbasis opgeleverd is, moet op basis van de twee voorgaande activiteiten het relevante deel hieruit geselecteerd worden. Als het niet mogelijk bleek om de CRUD-matrix als testbasis opgeleverd te krijgen, kan het testteam besluiten deze zelf samen te stellen op basis van de functionele specificaties. Dit is uiteraard niet wenselijk, maar een uiterste redmiddel.
- Iedere bewerking (C, R, U of D) die in de CRUD-matrix voorkomt is een aparte testsituatie die getest moet worden.

Testsituaties met betrekking tot integriteitsregels
De volgende activiteiten dienen uitgevoerd te worden:

- Verzamel de integriteitsregels over de geselecteerde gegevens.
  Dit zijn de regels die definiëren onder welke voorwaarden de *bewerkingen* op de gegevens geldig zijn of niet. Integriteitsregels zijn meestal gespecificeerd in de functionele specificaties, database-schema’s of in separate ‘business rules’.
- Pas decision coverage toe. Dat wil zeggen dat voor iedere integriteitsregel twee testsituaties afgeleid worden:
  – Ongeldig
    Er wordt niet voldaan aan de integriteitsregel. De bewerking is ongeldig en moet resulteren in een correcte foutafhandeling.
  – Geldig
    Er wordt wel voldaan aan de integriteitsregel. De bewerking is geldig en dient uitgevoerd te worden.

Uitgediept

Integriteitsregels dienen niet verward te worden met semantische regels, welke de voorwaarden definiëren waaronder de *waarde* van de gegevens zelf geldig zijn of niet. Bijvoorbeeld:

- “Bij het aanmaken van een bestelling moet de waarde van ‘aantal’ beneden de grens liggen die vastgelegd is in ‘product’.” is een semantische regel.
- “Het aanmaken van een factuur is alleen toegestaan als de betreffende bestelling reeds geaccordeerd is.” is een integriteitsregel.

Dus eerst bepaalt de integriteitsregel of de functie überhaupt toegestaan is. Daarna bepalen de semantische regels, of de invoergegevens die aan die functie aangeboden worden geldig zijn.

Voorbeeld uitwerking

De gegevenscyclustest wordt toegepast op een deelsysteem dat orders factureert en betalingen verwerkt. Het relevante deel van de CRUD-matrix is weergegeven in tabel 14.13.

	Artikel	Betalingsafspraak	Factuur	Grootboek
Beheren artikelen	C, R, U, D	-	-	…
Beheren betalingsafspraken	-	C, R, U, D	R	…
Beheren grootboek	-	-	R	C, R, U, D
Aanmaken factuur	R	R	C	U
Betaling-balie	-	-	C, U, D	U
Betaling-bank	-	-	U, D	U
…	…	…	…	…

Tabel 14.13: Voorbeeld van een uitgewerkte CRUD-matrix.

Voor dit deel van de CRUD-matrix bestaat één relevante integriteitsregel: Een betalingsafspraak mag niet verwijderd worden als er nog een factuur openstaat met de betreffende betalingsafspraak. Dit leidt tot twee testsituaties:
IR1-1: Verwijder (D) betalingsafspraak, terwijl er een factuur openstaat met de betreffende betalingsafspraak;
IR1-2: Verwijder (D) betalingsafspraak, zonder dat er een factuur openstaat met de betreffende betalingsafspraak;

Een verkorte overzichtelijke notatie voor dit soort testsituaties is bijvoorbeeld:

Testsituatie	Bewerking	Gegeven	Voorwaarde	Geldig J/N
IR1-1	D	betalingsafspraak	openstaande factuur	N
IR1-2	D	betalingsafspraak	geen openstaande factuur	J

De afkorting “IR” staat hierbij voor “integriteitsregel”.

2. Opstellen logische testgevallen

Stel 1 of meer logische testgevallen op, zodanig dat:

- ieder gegeven een volledige levensloop (beginnend met ‘C’ en eindigend met ‘D’) doorloopt;
- alle testsituaties uit de CRUD-matrix (iedere C, R, U en D) gedekt zijn;
- alle testsituaties van de relevante integriteitsregels gedekt zijn.

Zie ook paragraaf 14.3.7.

Een testgeval beschrijft dus een compleet scenario dat bestaat uit meerdere acties die elk een bewerking op een bepaald gegeven uitvoeren.

Voorbeeld uitwerking

Om het principe te illustreren worden de logische testgevallen voor de gegevens “Artikel” en “Betalingsafspraak” uitgewerkt in respectievelijk tabel 14.14 en tabel 14.15. De tabel beschrijft bij iedere rij welke functie gebruikt moet worden, welke bewerking (CRUD) op het betreffende gegeven hiermee afgedekt wordt en een korte toelichting met aanvullende informatie over de uit te voeren actie.

LTG-01: “Artikel”

Functie	CRUD	Actie / Toelichting
Beheren artikelen	C	Maak nieuw artikel ART-01
Beheren artikelen	R	Check ART-01
Aanmaken factuur	R	Maak factuur FCT-01 waarin artikel ART-01 voorkomt.
Beheren grootboek	-	Check FCT-01
Beheren artikelen	U	Wijzig ART-01 (bijv. prijs) in ART-01B
Beheren artikelen	R	Check ART-01B
Beheren grootboek	-	Check FCT-01 is ongewijzigd
Beheren artikelen	D	Verwijder ART-01B
Beheren artikelen	R	Check ART-01B (is verwijderd)

Tabel 14.14: Logisch testgeval bij een GCT voor het gegeven “Artikel”.

LTG-02: “Betalingsafspraak”

Functie	CRUD	Actie / Toelichting
Beheren betalingsafspraken	C	Maak nieuwe betalingsafspraak BAF-01
Beheren betalingsafspraken	R	Check BAF-01
Beheren betalingsafspraken	U	Wijzig BAF-01(bijv. periode) in BAF-01B
Beheren betalingsafspraken	R	Check BAF-01B
Aanmaken factuur	R	Maak factuur FCT-02 waarin afspraak BAF-01B staat
Beheren grootboek	-	Check FCT-02
Beheren betalingsafspraken	D	IR1-1. Foutafhandeling!
Beheren betalingsafspraken	R	Check BAF-01B bestaat nog steeds
Betaling-balie	-	Volledige betaling van FCT-02 zodat FCT-02 verwijderd wordt (niet meer open staat)
Beheren betalingsafspraken	D	IR1-2. Toegestaan
Beheren betalingsafspraken	R	Check BAF-01B is verwijderd

Tabel 14.15: Logisch testgeval bij een GCT voor het gegeven “Betalingsafspraak”.

Een “-” in de kolom “CRUD” betekent, dat de betreffende functie nodig is om een bepaalde actie uit te voeren, maar dat deze geen bewerking op het geteste gegeven uitvoert. Bijvoorbeeld: Bij LTG-01 wordt “Beheren grootboek” gebruikt om te kunnen controleren dat het juiste artikel op de factuur verschijnt, maar voert deze zelf geen bewerking uit op “Artikel”.

Bij LTG-02 wordt “Betaling-balie” gebruikt om factuur FCT-02 af te sluiten zodat aan integriteitsregel IR1-2 wordt voldaan, maar voert deze zelf geen bewerking uit op “Betalingsafspraak”.

3. Opstellen fysieke testgevallen

Bij het uitwerken van de logische testgevallen tot fysieke testgevallen worden de volgende details toegevoegd:

- (optioneel) Hoe de betreffende functie precies geactiveerd wordt. Meestal is dit voldoende duidelijk, maar soms is hier een niet-voor-de-hand-liggende opeenvolging van handelingen voor nodig;
- De in te voeren gegevens bij die functie. Als het logisch testgeval aangeeft dat een bepaald gegeven gewijzigd moet worden, dan moet het fysieke testgeval uitsluitsel geven over welk attribuut precies gewijzigd wordt in welke waarde;
- Een concrete beschrijving bij iedere resultaatverwachting van wat er bij een bepaald gegeven gecontroleerd moet worden;
- Extra acties die nodig zijn om volgende acties in het testgeval mogelijk te maken. Bijvoorbeeld het wijzigen van de systeemdatum of het uitvoeren van een bepaald batchproces om het systeem in een bepaalde benodigde status te krijgen.

4. Vaststellen uitgangssituatie

De GCT werkt typisch op overkoepelend systeemniveau, eventueel over verschillende systemen heen. Dat betekent dat een omvangrijke uitgangssituatie klaargezet moet worden die volledig en consistent over alle systemen heen is.

Met name moet het volgende geregeld zijn:

- Alle benodigde databases voor alle betrokken systemen waarin alle gegevens consistent gevuld zijn;
- Een configuratie (eventueel een netwerk) waarin alle benodigde systemen gekoppeld zijn en waarin alle benodigde gebruikers gedefinieerd zijn met de noodzakelijke toegangsrechten.

Een dergelijke uitgangssituatie benadert de productiesituatie en is complex om op te bouwen. Bij voorkeur wordt gebruikt gemaakt van een reeds bestaande ‘real life testomgeving’. Zie ook paragraaf 6.6.2 “Definiëren centrale uitgangssituatie(s)”.

In het bijzonder moet gelet worden op gegevens in de uitgangssituatie die slechts beperkte tijd geldig zijn. Bij aanvang van iedere testuitvoering zal gecontroleerd moeten worden of deze tijdafhankelijke gegevens nog geldig zijn en of op basis hiervan wijzigingen in de uitgangssituatie aangebracht moeten worden.

14.4.8 Procescyclustest (PCT)

Omschrijving

De procescyclustest is een techniek die vooral wordt toegepast bij het testen van het kwaliteitsattribuut inpasbaarheid (integratie tussen de administratieve organisatie en het geautomatiseerde informatiesysteem). De testbasis moet gestructureerde informatie bevatten over het gewenste systeemgedrag in de vorm van paden en beslispunten De procescyclustest wijkt op een aantal punten af van de meeste andere testontwerptechnieken:

- De procescyclustest is geen ontwerptest maar een structuurtest. De testgevallen komen immers voort uit de structuur van de procedureflow en niet uit de ontwerpspecificaties.
- De resultaatvoorspelling bij de procescyclustest is simpel: het fysieke testgeval moet uitvoerbaar zijn. Hiermee is impliciet gecontroleerd dat de individuele acties uitvoerbaar zijn. Er wordt, in tegenstelling tot andere testontwerptechnieken, geen expliciete voorspelling van het resultaat gemaakt en dus hoeft daar ook niet op te worden gecontroleerd.

De procescyclustest richt zich op het afdekken van de variaties in het procesverloop. De basistechniek die hierbij wordt gebruikt, is:

- Paden: testmaat-2

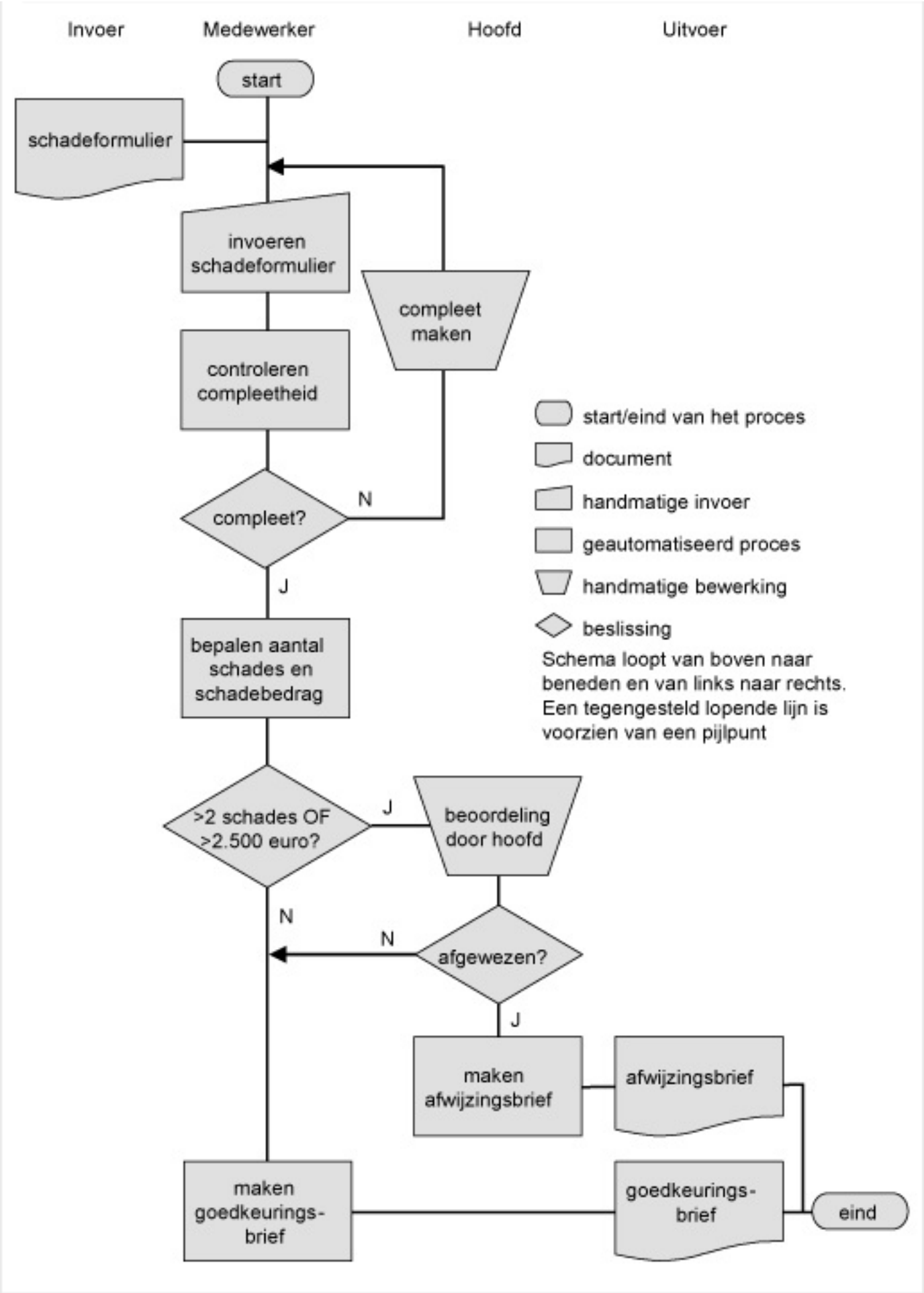
Variaties op de procescyclustest kunnen worden gecreëerd door het toepassen van de basistechniek:

- Paden: testmaat-1, testmaat-3 en hoger
  Hiermee kunnen paden respectievelijk *lichter* of *zwaarder* worden getest.

Aandachtspunten bij de stappen

In deze paragraaf wordt de procescyclustest stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 “Inleiding”) als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat bij iedere stap laat zien hoe deze techniek in zijn werk gaat. In het voorbeeld wordt voor het weergeven van een detailprocesschema een bepaalde tekentechniek gebruikt. Aangezien er geen uniforme afspraken bestaan over het gebruik van de techniek en het gebruik van de symbolen, is de betekenis van de symbolen naast het schema vermeld.

Voorbeeld uitwerking



Figuur 14.17: Deel van het detailprocesschema "schadeafhandeling".

Bij de schadeafdeling komen schadeformulieren binnen. Nadat een medewerker van deze afdeling de schadegegevens het systeem heeft ingevoerd, wordt een proces gestart om de compleetheid van de gegevens te controleren. Bij incompleetheid neemt de medewerker contact op met de verzekerde, waarna de gegevens opnieuw worden ingevoerd. Als de gegevens compleet zijn wordt een proces gestart om het schadebedrag en het aantal

schadeaangiften van de betreffende verzekerde van het afgelopen jaar te bepalen. Als het schadebedrag hoger is dan €2.500, of als in het afgelopen jaar meer dan 2 schadeaangiften zijn gemeld, dan moet het hoofd van de afdeling beoordelen of het schadebedrag wel of niet wordt uitgekeerd. Bij afwijzing door het hoofd, wordt een afwijzingsbrief gemaakt, bij goedkeuring wordt een goedkeuringsbrief gemaakt. Bij 2 of minder schadeaangiften in het afgelopen jaar en bij een schadebedrag van €2.500 of lager wordt altijd een goedkeuringsbrief gemaakt.

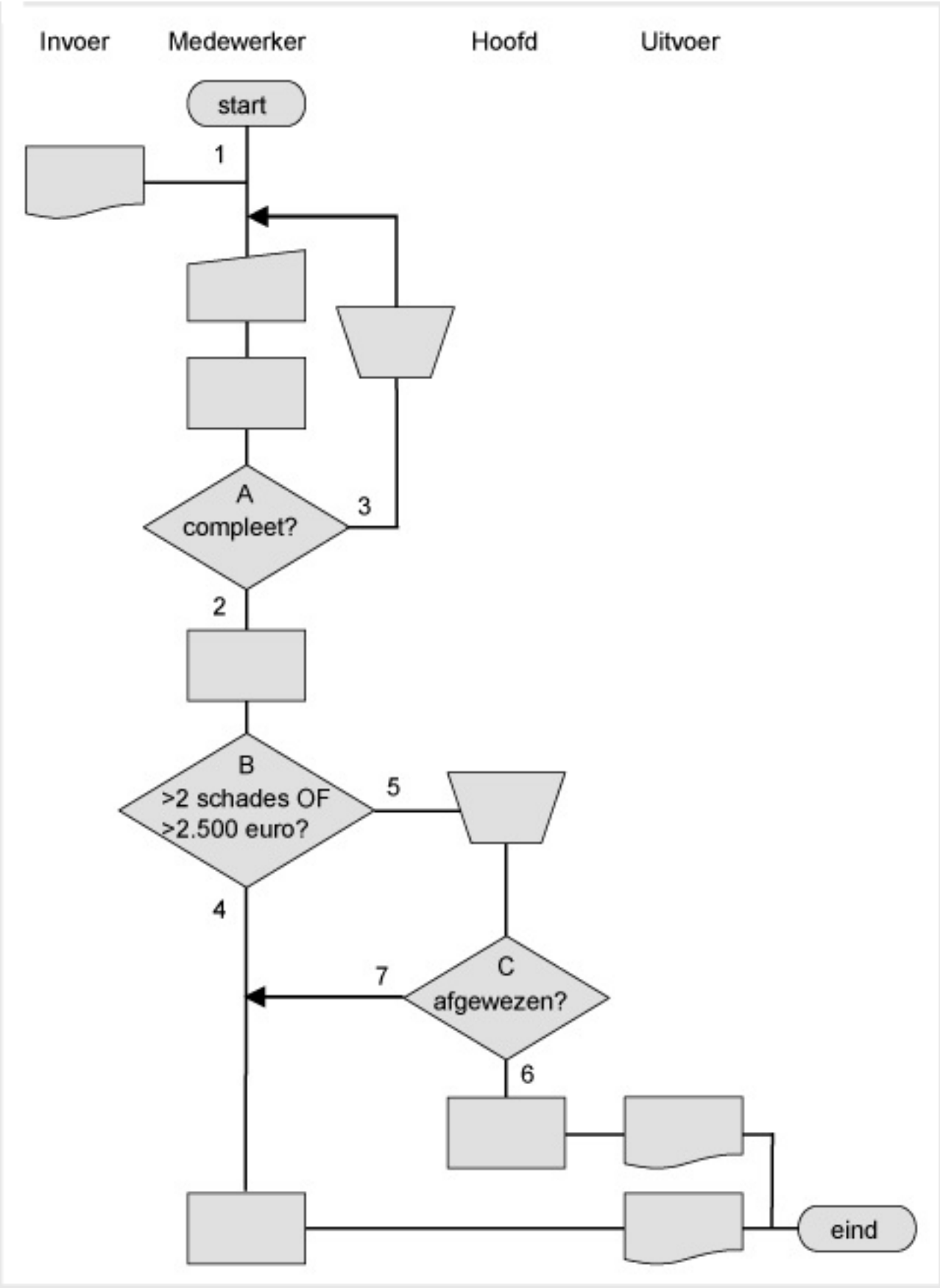
Hieronder is per stap uitgewerkt hoe de procescyclustest wordt toegepast op dit proces:

1. Identificeren testsituaties

Voor het kunnen toepassen van de procescyclustest is een processchema nodig. Dit schema moet, naast een start- en een eindpunt, beslispunten en paden bevatten. Bevat de testbasis al een schema, dan is het voor de overzichtelijkheid vaak handig deze ‘uit te kleden’ tot een schema dat alleen bovenstaande aspecten bevat. Als in de testbasis geen schema aanwezig is, zal de tester zelf de beslispunten en paden uit de testbasis moeten destilleren om een schema te kunnen opstellen. Vervolgens worden uit het schema de testsituaties afgeleid volgens de in paragraaf 14.3.2 beschreven techniek bij de dekkingsvorm “paden”.

Voorbeeld

De tester heeft in het “schadeafhandelings” detailprocesschema de, voor de procescyclustest, overbodige informatie verwijderd en de paden genummerd. Het resultaat is weergegeven in figuur 14.18.



Figuur 14.18: Vereenvoudigde weergave van het processchema.

Toepassen van de dekkingsvorm “paden met testmaat-2” leveren de volgende testsituaties (padcombinaties) op:

A: 1-2; 1-3; 3-2; 3-3
B: 2-4; 2-5
C: 5-6; 5-7

2. Opstellen logische testgevallen

Het opstellen van de logische testgevallen valt in twee activiteiten uiteen:

- samenstellen set logische testgevallen;
- per logisch testgeval de opeenvolgende acties beschrijven.

Bij het samenstellen van de set logische testgevallen moeten alle testsituaties worden afgedekt. Een testgeval wordt gedefinieerd door op een bepaalde manier door het proces te lopen van “Start” tot “Eind” (zie paragraaf 14.3.2). De tester is vrij om te kiezen op welke manier het proces wordt doorlopen, mits alle testsituaties minimaal 1x worden afgedekt.

Voorbeeld uitwerking

Voor het proces schadeafhandeling komt de tester tot de volgende set logische testgevallen:

TG-1 = 1-2-4
TG-2 = 1-2-5-6
TG-3 = 1-3-3-2-5-7

Ga zelf na, dat de volgende set ook één van de mogelijke sets is:

TG-1 = 1-3-3-2-4
TG-2 = 1-2-5-6
TG-3 = 1-2-5-7

Was er volgens testmaat-1 afgeleid, dan had dat bijvoorbeeld tot de volgende set logische testgevallen geleid (ga zelf na dat ook hier meerdere sets mogelijk zijn):

TG-1 = 1-2-4
TG-2 = 1-2-5-6
TG-3 = 1-3-2-5-7

De logische testgevallen moeten vervolgens worden uitgeschreven. Dit betekent dat er voor ieder testgeval een rij opeenvolgende acties moet worden beschreven, zodanig dat door het uitvoeren van die acties alle testsituaties uit het testgeval worden bewandeld. Dit is een activiteit die de nodige inventiviteit vereist en is daarom vrij moeilijk in algemene termen te beschrijven. Zie het voorbeeld voor een concrete invulling.

Voorbeeld uitwerking

De tester heeft voor de concrete invulling van TG-3 (1-3-3-2-5-7) de volgende acties beschreven:

A3-1	Aanmaken schadeformulier	(verzekerde)
A3-2	Gegevens schadeformulier het systeem invoeren (incompleet)	(medewerker)
A3-3	Proces “controle op compleetheid” starten	(medewerker)
A3-4	Contact met verzekerde opnemen om gegevens te completeren	(medewerker)
A3-5	Gegevens schadeformulier inclusief één aanvullend gegeven het systeem invoeren (incompleet)	(medewerker)
A3-6	Proces “controle op compleetheid” starten	(medewerker)
A3-7	Contact met verzekerde opnemen om gegevens te completeren	(medewerker)
A3-8	Gegevens schadeformulier inclusief aanvullende gegevens het systeem invoeren (compleet, schadebedrag > €2.500)	(medewerker)
A3-9	Proces “controle op compleetheid” starten	(medewerker)

A3-10	Proces “bepalen aantal schades en schadebedrag” starten	(medewerker)
A3-11	Beoordelen schademelding	(hoofd)
A3-12	Schademelding wordt geaccepteerd	(hoofd)
A3-13	Proces “brief maken” (goedkeuringsbrief) starten	(medewerker)
A3-14	Einde proces	

3. Opstellen fysieke testgevallen

Naast de eerder genoemde verschillen met de andere testontwerptechnieken is er nog een verschil te noemen. Bij de testuitvoering is bij de procescyclustest meer nodig dan alleen de technische testinfrastructuur waar het geautomatiseerde deel van het informatiesysteem op draait. De handmatige procedures moeten namelijk veelal worden uitgevoerd door verschillende soorten medewerkers, wat betekent dat er bij de testuitvoering meerdere testers nodig zijn die een bepaald rollenspel moeten spelen. Het is uiteraard ook mogelijk de test uit te laten voeren door één tester die over meerdere user-id’s beschikt en tijdens de testuitvoering meerdere keren in- en uitlogt. Tenslotte zitten de benodigde gegevens slechts voor een deel in de database van het geautomatiseerde deel van het informatiesysteem maar voor het overige daarbuiten, bijvoorbeeld in de vorm van ingevulde formulieren. Ook dat is een verschil met de overige testontwerptechnieken.

Bij het opstellen van de fysieke testgevallen wordt een fysieke invulling van de logische testgevallen gegeven. Hierbij dienen de beschreven acties als uitgangsbasis.

Voorbeeld uitwerking

De tester heeft de fysieke invulling van TG-3 (1-3-3-2-5-7) als volgt beschreven:

Verzekerde:
A3-1 Maak schadeformulier aan met de volgende gegevens:

Naam	: Janssen, J.
Adres	: Amsterdamstraat 7, Utrecht
Datum schade	: <leeg>
Omschrijving schade	: <leeg>
Oorzaak schade	: diefstal uit woning
Schadebedrag	: €4.250

Medewerker (MEDEW_01):

A3-2	Gegevens schadeformulier het systeem invoeren (zonder “datum schade” en zonder “omschrijving schade”)
A3-3	Proces “controle op compleetheid” starten (formulier is incompleet)
A3-4	Contact met verzekerde opnemen voor het verkrijgen van de “datum schade”
A3-5	Gegevens schadeformulier het systeem invoeren (met “datum schade” is 1 december 2006)
A3-6	Proces “controle op compleetheid” starten (formulier is incompleet)
A3-7	Contact met verzekerde opnemen voor het verkrijgen van de “omschrijving schade”
A3-8	Gegevens schadeformulier het systeem invoeren (met “omschrijving schade” is “gestolen collier”)
A3-9	Proces “controle op compleetheid” starten (formulier is compleet)
A3-10	Proces “bepalen aantal schades en schadebedrag” starten (schadebedrag > 2.500, dus beoordeling door hoofd)

Hoofd (HOOFD_01):

A3-11	Beoordelen schademelding

A3-12 Schademelding accepteren

Medewerker (MEDEW_01):

A3-13 Proces “brief maken” starten (goedkeuringsbrief wordt aangemaakt)
A3-14 Einde proces

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

Voorbeeld uitwerking

Voor het kunnen uitvoeren van de testgevallen uit het schadevoorbeeld moeten:

- in de database de verzekeringsgegevens van de verzekerden aanwezig zijn;
- de user-id’s van een medewerker en het afdelingshoofd met relevante autorisatie(s) in het systeem bekend zijn;
- de schadeformulieren zijn ingevuld.

Als voorbeeld wordt hier de uitgangssituatie voor TG-3 gegeven.

Verzekeringsgegevens:

Naam	: Janssen, J.
Adres	: Amsterdamstraat 7, Utrecht
Aantal schades 2006	: 2

User-id’s:
De user-id’s “MEDEW_01” en “HOOFD_01” moeten met bijbehorende autorisatie(s) in het systeem aanwezig zijn.

Schadeformulier:

Naam	: Janssen, J.
Adres	: Amsterdamstraat 7, Utrecht
Datum schade	: <leeg>
Omschrijving schade	: <leeg>
Oorzaak schade	: diefstal uit woning
Schadebedrag	: €4.250

14.4.9 Real life test (RLT)

Omschrijving

Bij de real life test is het niet de bedoeling om het systeemgedrag in afzonderlijke situaties te testen, maar om het realistisch gebruik van het systeem statistisch verantwoord te simuleren. Deze test richt zich vooral op kenmerken als bruikbaarheid, connectiviteit, continuïteit en performance van het te testen systeem. Veel fouten die met een real life test worden gevonden houden verband met het resourcegebruik van een systeem:

- ‘crashen’ van transacties na langdurig gebruik;
- ‘crashen’ van transacties die in een bepaalde volgorde worden uitgevoerd;
- onvoldoende responsetijden en snelheid van verwerking;
- onvoldoende geheugen- of opslagruimte beschikbaar;
- onvoldoende capaciteit van randapparatuur en datacommunicatienetwerk.

Om te kunnen testen of het systeem bestand is tegen het realistisch gebruik ervan, moet dat gebruik op een of andere wijze zijn gespecificeerd. Dit is tevens de testbasis en wordt in dit verband vaak ‘profile’ genoemd. De twee bekendste

zijn:

- Operational profiles
  Simuleren van het realistisch gebruik van het systeem, door een *opeenvolging van transacties* uit te voeren, die statistisch verantwoord is samengesteld.
- Load profiles
  Simuleren van een realistische belasting van het systeem in termen van *aantallen* gebruikers en/of transacties.

De basistechniek die hierbij wordt gebruikt, is:

- Statistisch verantwoorde simulatie

Voor een nadere toelichting op deze profiles en het simuleren ervan wordt naar paragraaf 14.3.8 “Statistisch gebruik: Operational profiles en Load profiles” verwezen.

In de praktijk wordt bij een real life test vaak een mix van deze profiles gebruikt. Hierbij wordt dan een bepaalde belasting van het systeem gesimuleerd door het uitvoeren van realistische scenario’s.

Een profile wordt gebruikt bij de uitwerking van één of meer testdoelen van de real life test. Voorbeelden van testdoelen zijn:

- Testen bij een normaal of gemiddeld gebruik
  Het doel is hier om te onderzoeken of de beschikbare systeemresources voldoende zijn voor de gebruikelijke omstandigheden. Vaak betreft het hier een test met een gemiddeld aantal gebruikers die interactief werk verrichten, overzichten draaien en een aantal kleine batchfunctionaliteiten uitvoeren.
- Testen bij intensief gebruik
  Het doel is hier om te onderzoeken of er voldoende systeemresources zijn voor zelfs de zwaarste, maar wel realistische, omstandigheden. Vaak betreft het hier een test met een maximaal aantal gebruikers die interactief werk verrichten (piekbelasting) of een test waarbij bepaalde transacties vaak en langdurig worden uitgevoerd.
- Meten van breekpunt (stress testen)
  Het doel is hier om te onderzoeken wat de maximale belasting is waaronder het systeem nog acceptabel presteert. Vaak betreft het hier een test met een toenemend aantal (gesimuleerde) gebruikers.
- Testen van dagbatches
  Het doel is hier om te onderzoeken of de beschikbare systeemresources voldoende zijn voor de combinatie van een normaal aantal interactieve gebruikers met het gelijktijdig uitvoeren van relatief zware batchjobs.
- Testen van nachtbatches
  Het doel is hier om te onderzoeken of zowel de beschikbare systeemresources als beschikbare tijd voldoende zijn voor het (‘s nachts / in het weekend) uitvoeren van zware batchjobs.

Het uitvoeren van de real life test is doorgaans ingewikkelder dan bij andere tests. In een omgeving waarin het aantal eindgebruikers niet al te groot is, kan men gedurende een weekeinde iedereen laten overwerken en de testscripts volgens een tevoren vastgelegde detailpanning laten uitvoeren. Maar steeds vaker wordt gebruik gemaakt van tools die op verschillende manieren een realistische belasting simuleren. Dit zijn tools die bijvoorbeeld het aantal gebruikers simuleren door het creëren van virtuele gebruikers of tools die een zekere belasting van het back-end van het systeem simuleren door transacties rechtstreeks via de databasemanagement interface aan te bieden (dus zonder gebruikmaking van het front-end of netwerk).

Er moet van te voren goed worden bepaald wat en hoe men gedurende de real life test gaat meten. Soms belast het meten op zich ook het systeem wat tot vervorming van de resultaten kan leiden. Anderzijds moet men voldoende gegevens hebben om achteraf een goede analyse uit te kunnen voeren.

Het beoordelen van het resultaat van een real life test kan soms lastig zijn.

Het komt nogal eens voor dat tests niet reproduceerbaar zijn, omdat vaak fouten worden gevonden die worden veroorzaakt door onvoldoende geheugen, langdurig gebruik enzovoort. Dit soort fouten (denk aan memory leaks) is moeilijk reproduceerbaar, omdat er vrijwel altijd invloeden van buiten meespelen die niet of nauwelijks onder controle te krijgen zijn, zoals het geheugenmanagement van het operating system. Bij het opsporen van de oorzaken van eventuele fouten kunnen logging- en monitoringfaciliteiten worden gebruikt.

Aandachtspunten bij de stappen

Voor de real life test is het opstellen of achterhalen van correcte profiles de belangrijkste stap. Dit vindt plaats in stap 1 “Identificeren testsituaties”. De exacte inhoud van de testgevallen is minder relevant dan voor de meeste andere testontwerptechnieken. Het belangrijkste criterium is dat de realiteit qua omvang en frequentie van gebruik zo dicht mogelijk wordt benaderd. Dit houdt in dat het meestal niet zinvol is om logische testgevallen te maken.

Vaak kan direct al de fysieke invulling worden gegeven om de gewenste testsituaties af te dekken.

1. Identificeren testsituaties

De profiles zijn te zien als de te testen testsituaties. Deze geven op hoofdlijnen aan welke acties (functies) gedurende een bepaalde periode worden uitgevoerd en het aantal actieve gebruikers. Men kan hierbij denken aan een aantal dagcycli, bijvoorbeeld een minimale, gemiddelde en maximale. Een dagcyclus bestaat bijvoorbeeld uit inloggen, intensief gebruik, minder intensief gebruik tijdens de lunch, intensief gebruik, uitloggen, back-up en dagbatches. Daarnaast kunnen er ook vergelijkbare week-, maand- en jaarcycli en specifieke processen zoals back-up en recovery bestaan. Er zijn verschillende manieren om aan de benodigde informatie voor het opstellen van een operational profile, load profile of een mix daarvan te komen. Een aantal manieren in willekeurige volgorde:

- uit huidige systeem of release het profile afleiden;
- overnemen van een bestaande profile van een systeem met vergelijkbare functionaliteit;
- overnemen van een bestaande profile met vergelijkbare belasting van de systeemresources;
- bijhouden van gegevens (logging) over hoe vaak iedere functie wordt gebruikt;
- met behulp van specifieke tools (monitors) de belasting van het systeem meten;
- interviewen van gebruikers, waarbij de kernvraag is: “Welke transacties voer je hoe vaak en wanneer uit, of welke transacties verwacht je op het nieuwe systeem uit te gaan voeren?”

Een belangrijke overweging bij het opstellen van een profile is de mate van detail. Een meer omvangrijke en gedetailleerde profile zal de realiteit uiteraard beter weergeven, maar zal tevens leiden tot een toename van de testinspanning voor het specificeren en realiseren van testgevallen en uitvoeren van de real life test. In de profile moet ervoor worden gezorgd dat alle systeemresources realistisch worden gebruikt. Het is zinloos een significant zwaarder gebruik te simuleren dan in realiteit gebruikelijk is, omdat de uitslag van zo’n test niets zegt. Als het systeem bijvoorbeeld te traag is onder die omstandigheden wil het niet zeggen dat het systeem niet zal voldoen. Als het systeem niet te traag is, zegt dat alleen dat het systeem overgeconfigureerd is, zonder dat duidelijk wordt hoeveel. Een significant zwaarder gebruik simuleren heeft uiteraard wel weer zin als het doel van de test is om te bepalen wat de maximale belasting is waaronder het systeem nog acceptabel presteert.

Tip

Voor het nader detailleren van een profile kunnen de volgende activiteiten worden uitgevoerd. Hierin is een aantal van bovengenoemde manieren om aan de informatie voor het opstellen van een profile te komen, verwerkt:

- Identificeren gebruikersgroepen
  Het identificeren van de gebruikersgroepen kan bijvoorbeeld gebeuren aan de hand van AO-beschrijvingen, analyse van de doelgroepen waar eventuele output voor bestemd is of beveiligingsoverzichten op functieniveau. Gebruikersgroepen zijn veelal sterk gerelateerd aan organisatorische functies en taken. Ten behoeve van de backup en recovery processen dienen de systeembeheerders niet te worden vergeten.
- Overzicht taken per gebruikersgroep
  Een taak is gedefinieerd als een aantal opeenvolgende handelingen die een gebruiker uitvoert in dialoog met het systeem gericht op een bepaald doel. De functionele specificaties zijn wellicht de belangrijkste bron van informatie voor deze activiteit. Alternatieven zijn de bedrijfsprocessen, AO-beschrijvingen, de gebruikershandleiding, een prototype van het systeem, en eventueel een vorige versie van het informatiesysteem. Het is aan te raden om deze activiteit uit te voeren door middel van direct contact (interviews) met gebruikers. Het betrekken van gebruikers tijdens deze activiteit is veelal verhelderend en levert allerlei aanvullende informatie op.
- Bepaal de frequentie per taak per tijdseenheid
  Gegevens met betrekking tot de frequentie zijn vaak al aanwezig uit voorgaande systemen met vergelijkbare functionaliteit. Zo niet, dan dienen deze te worden verzameld bijvoorbeeld door in het huidige systeem bij te houden hoe vaak een bepaalde functie wordt gestart dan wel is uitgevoerd, of door het interviewen van gebruikers. Het is belangrijk hierbij in het achterhoofd te houden dat het gaat om schattingen van de frequentie en niet om exacte metingen.

- Bepaal de relatieve frequentie
  De relatieve frequentie wordt bepaald door het voorkomen van een bepaalde taak in een bepaalde tijdseenheid te delen door het totaal aantal voorkomens van alle taken die in die periode worden uitgevoerd. De testgevallen worden gerealiseerd in overeenstemming met de taken en de relatieve frequentie.

Voorbeeld

Bij een organisatie die banktransacties verwerkt voor diverse banken worden, bij normaal gebruik, per uur 275.000 transacties verwerkt. Deze zijn als volgt over de transactietypen verdeeld (zie tabel 14.16):

Transactietype	Frequentie/uur	Relatieve frequentie
Betaalautomaattransacties	150.000	0,55
Incasso opdrachten	90.000	0,33
Geldautomaat transacties	20.000	0,07
Acceptgiro betalingen	15.000	0,05
Totaal	**275.000**	**1,0**

Tabel 14.16: Voorbeeld van een frequentieverdeling van transactietypen.

De testgevallen worden gerealiseerd in overeenstemming met de taken en de relatieve frequentie. Voor het voorbeeld betekent dit de volgende verdeling over de testgevallen: 55% betaalautomaat transacties, 33% incasso opdrachten, 7% geldautomaattransacties en 5% acceptgiro betalingen.

Mogelijke doelen van de test zijn:

- Wat is het maximale aantal transacties dat per seconde kan worden verwerkt?
- Wat is de gemiddelde doorlooptijd van één transactie bij normaal of intensief gebruik en valt deze binnen de afgesproken grenzen?
- Is het systeem bestand tegen langdurig ononderbroken gebruik?

2. Opstellen logische testgevallen

Er wordt vaak rechtstreeks overgegaan naar het opstellen van fysieke testgevallen.

3. Opstellen fysieke testgevallen

Voor de real life test is de exacte inhoud van de fysieke testgevallen minder relevant dan voor de meeste andere testontwerptechnieken. Het enige criterium is dat de realiteit qua omvang en frequentie van gebruik zo dicht mogelijk moet worden benaderd. Dit klinkt eenvoudiger dan het in werkelijkheid is. Er moet immers goed worden nagedacht over *hoe* een bepaald gebruik of een bepaalde belasting van het systeem kan worden gerealiseerd of gesimuleerd. Hiernaast moeten voor sommige tests testgevallen worden verzameld of gemaakt, die dan bij de testuitvoering worden uitgevoerd. Terwijl voor het uitvoeren van andere tests al *vooraf* een bepaalde vulling in het systeem aanwezig moet zijn.

Het maken van testgevallen kan bijvoorbeeld door het fysiek uitwerken van gebruiksscenario's, of als er al een operationeel systeem is door het 'aftappen' van een representatieve testset.

Voor het testen van bepaalde aspecten van het systeemgebruik kunnen de testgevallen worden gerealiseerd door een dagproductie (na bewerking) als real life invoer klaar te zetten. Bij het gebruik van productiedata dient overigens wel te worden gedacht aan privacyaspecten! Verder moeten de uitgewerkte gebruiksscenario's en acties die onderdeel zijn van de real life test zo realistisch mogelijk worden verdeeld over de gebruikers (testers) die meewerken aan de test.

Het gebruik van een tool om bijvoorbeeld gebruikers of transacties te simuleren betekent niet dat er geen gebruiksscenario's en dergelijke hoeven te worden uitgewerkt. Immers, ook bij gebruik van een tool vormen deze gebruiksscenario's de basis voor de test. Aanvullend zal het tool zo geprogrammeerd of ingesteld moeten worden dat het de gebruiksscenario's kan uitvoeren.

Voorbeeld uitwerking

Bij het voorbeeld van de transactieverwerking is voor de betaalautomaattransacties de volgende fysieke invulling gegeven (zie tabel 14.17):

Nr	Transactietype	Parameters*	Doel en verwachting
1	Betaalautomaat transacties	Aantal betaaltransacties in 90 minuten op laten lopen van 5tr/sec naar 450 tr/sec.	Bepalen breekpunt (stresstest). Verwachting: ≥ 350 tr/sec.
2	Betaalautomaat transacties	42 tr/sec (gedurende 5 minuten)	Bepalen doorlooptijd van een betaling bij normaal gebruik.
3	Betaalautomaat transacties	120 tr/sec (gedurende 5 minuten)	Bepalen doorlooptijd van een betaling bij intensief gebruik.
Bij testgevallen 2 en 3 bestaat de vaste systeembelasting, naast de betaalautomaat transacties, uit 25 incasso's/sec, 5 geldopnames/sec en 4 acceptgirobetalingen/sec. De betaling moet voor 95% < 5 sec, voor 99% < 7 sec en voor 100% < 10 sec worden uitgevoerd.			
4	Betaalautomaat transacties	42 tr/sec (gedurende 7 dagen)	Testen of het systeem bij langdurig gebruik niet 'crasht'.
* De betalingstransacties bedragen gemiddeld €100,= en zijn verdeeld over 4 banken.			

Tabel 14.17: Fysieke testgevallen voor de betaalautomaattransacties.

4. Vaststellen uitgangssituatie

Het creëren van een correcte uitgangssituatie voor een real life test behoeft, net als bij de meeste andere tests, vaak veel aandacht. Maar bij de real life test komen daar vaak nog extra aandachtspunten bij:

- testomgeving moet representatief zijn voor de productiesituatie;
- er wordt gebruik gemaakt van omvangrijke bestanden;
- er zijn veel gebruikers (testers) die testen;
- er moet beschikt kunnen worden over een 'echt' netwerk.

Voor meer informatie wordt verwezen naar paragraaf 6.6.2 "Definiëren centrale uitgangssituatie(s)", waarin dit onderwerp uitvoerig wordt behandeld.

14.4.10 Semantische test (SEM)

Omschrijving

De semantische test (SEM) behoort samen met de syntactische test tot de *validatietests* waarmee de *geldigheid van de invoergegevens* getest wordt. In de praktijk wordt de semantische test vaak gecombineerd uitgevoerd met de syntactische test (zie paragraaf 14.4.11).

De testbasis bestaat uit de semantische regels die specificeren waar een gegeven aan moet voldoen om als geldige invoer door het systeem geaccepteerd te worden. Semantische regels hebben te maken met de *relatie tussen gegevens*. Deze relaties kunnen liggen tussen de gegevens binnen een scherm, tussen gegevens op verschillende schermen en tussen invoergegevens en reeds aanwezige gegevens in de database. Semantische regels kunnen in verschillende documenten vastgelegd zijn, maar meestal zijn ze beschreven in:

- Functionele specificaties van de betreffende functie of invoerscherm
- De 'business rules' die overkoepelend voor alle functies gelden

Tip

Als de semantische regels de voorwaarden beschrijven om aan beveiligingseisen te voldoen, kan de SEM dus ook toegepast worden voor de testvorm "beveiligingstest". Met de semantische test kunnen ook gebruiksvriendelijkheidsaspecten getest worden, door de meldingen die optreden bij ongeldige situaties te beoordelen op:

- Zijn ze begrijpelijk en ondubbelzinnig?
- Bieden ze heldere aanknopingspunten hoe de ongeldige situatie opgelost kan worden?

Aangezien de semantische regels gespecificeerd kunnen worden als beslispunten die bestaan uit samengestelde condities, wordt voor de semantische test één van de dekkingsvormen gekozen op het gebied van beslispunten. De default keuze voor de semantische test is:

- modified condition/decision coverage.

Op eenvoudige wijze kunnen varianten gerealiseerd worden door deze te vervangen door:

- condition/decision coverage, voor een lichte variant;
- multiple condition coverage, voor een zware variant.

Aandachtspunten bij de stappen

Ook voor de SEM worden in principe de generieke stappen (zie 14.4.1 “Inleiding”) uitgevoerd. Echter de opbouw van een semantische test is vrij triviaal: Iedere semantische regel wordt afzonderlijk getest. Iedere regel leidt tot één of meer testsituaties en iedere testsituatie leidt in het algemeen tot één testgeval.

Daarom beperkt deze paragraaf zich tot het toelichten van de eerste stap “identificeren testsituaties”. Deze zal met een voorbeeld toegelicht en uitgediept worden.

1. Identificeren testsituaties

Een semantische regel die de geldigheidsvoorwaarden beschrijft, kan in het algemeen als volgt uitgeschreven worden:

ALS (semantische regel)	DAN	geldige invoer c.q. verwerking
	ANDERS	foutmelding

In het geval dat de semantische regel de *on*geldige situaties beschrijft waarbij een foutmelding op moet treden, wordt dit:

ALS (semantische regel)	DAN	foutmelding
	ANDERS	geldige invoer c.q. verwerking

De semantische regel is een beslispunt dat bestaat uit één of meer condities verbonden door EN en OF. Bij een enkelvoudige conditie zijn er slechts twee testsituaties, één voor geldige invoer en één voor ongeldige invoer. Bij samengestelde condities worden de testsituaties afgeleid door het toepassen van modified condition/decision coverage (MCDC), zoals toegelicht in paragraaf 14.3.3.

Voorbeeld uitwerking

Stel dat de volgende semantische controle gespecificeerd is: “ALS klant in Nederland woont EN (postcode voldoet niet aan Nederlands formaat OF landcode is niet gelijk aan 31) DAN resulteert dit in een foutmelding.” Na toepassing van MCDC ontstaat:

B1 A EN (B OF C)	1 foutmelding	0 geldige invoer
A: klant in NL	1 1 0 (1)	0 1 0 (3)
B: postcode niet in NL	~~1 1 0~~	1 0 0 (4)
C: landcode ≠ 31	1 0 1 (2)	~~1 0 0~~

Uitgediept

In de praktijk kan het voorkomen dat semantische regels beschreven zijn in de vorm “ALS gegeven-X voldoet aan voorwaarde-A DAN moet ook voldaan worden aan de voorwaarden …”

Hier is de valkuil dat het lijkt alsof de semantische regel alleen bestaat uit de conditie “ALS gegeven-X voldoet aan voorwaarde-A”. Dat is echter niet zo. Ook alles wat achter de “DAN” staat, beschrijft de condities waaraan voldaan moet worden. Feitelijk is deze schrijfwijze van de semantische regel een voorbeeld van de zogenaamde “imply-operator” in de Booleaanse algebra. De waarheidstabel voor deze operator – die weergegeven wordt met het

symbool “→” - is:

A	B	A → B
1	1	1
1	0	0
0	1	1
0	0	1

Nu kan een conditie die beschreven is met de imply-operator eenvoudig omgezet worden naar een samengestelde conditie met dezelfde waarheidstabel:

“A → B” is equivalent met “(NIET A) OF B”

Op de hieruit ontstane samengestelde conditie - die uitsluitend de operatoren EN, OF en NIET bevat - kan zonder problemen de basistechniek modified condition/decision coverage toegepast worden.

Het onderstaande voorbeeld licht dit verder toe.

Stel dat de volgende semantische regel gespecificeerd is:

“Als code_bijdrage = V DAN moet code_dienstverband = F EN leeftijd ≥ 55”

Hierin is een imply-operator toegepast, waardoor de regel er feitelijk uitziet als:

“code_bijdrage = V → (code_dienstverband = F EN leeftijd ≥ 55)”

Deze kan omgezet worden naar de volgende samengestelde conditie:

”(NOT code_bijdrage = V) OF (code_dienstverband = F EN leeftijd ≥ 55)”

ofwel

“code_bijdrage ≠ V OF (code_dienstverband = F EN leeftijd ≥ 55)”

Toepassen van de basistechniek MCDC levert de volgende vier testsituaties op:

B1 **A OF (B EN C)**	**1** **geldige invoer**	**0** **foutmelding**
A: code_bijdrage ≠ V	1 1 0 (1)	0 1 0 (3)
B: code_dienstverband = F	0 1 1 (2)	0 0 1 (4)
C: leeftijd ≥ 55	~~0 1 1~~	~~0 1 0~~

2. Opstellen logische testgevallen

De testsituaties uit stap 1 zijn direct de logische testgevallen.

Voorbeeld uitwerking

Het uitwerken van de vier testsituaties uit ons voorbeeld levert direct de vier logische testgevallen:

Testgevallen/Testsituaties	B1-1	B1-2	B1-3	B1-4
Klant	in NL	in NL	niet in NL	in NL
Postcode	niet in NL	in NL	niet in NL	in NL
Landcode	31	≠ 31	31	31
Resultaatverwachting	Foutmelding	Foutmelding	OK	OK

3. Opstellen fysieke testgevallen

Geen bijzonderheden.

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

14.4.11 Syntactische test (SYN)

Omschrijving

De syntactische test (SYN) behoort samen met de semantische test tot de *validatietests* waarmee de *geldigheid van de invoergegevens* getest wordt. Hiermee wordt vastgesteld in hoeverre het systeem bestand is tegen ongeldige of 'onzinnige' invoer die al of niet moedwillig aan het systeem aangeboden wordt.

Overigens wordt met deze test ook de geldigheid van *uitvoergegevens* getest.

Validatietests houden zich bezig met *rubrieken* die niet verward moeten worden met *gegevens*. Een invoerscherm of andere willekeurig interface bevat rubrieken die met de invoerwaarden gevuld worden. Als de rubrieken geldige waarden bevatten, zal het systeem deze verwerken en daarmee in het algemeen bepaalde gegevens in het systeem aanmaken of wijzigen.

De testbasis voor de syntactische test bestaat uit de syntactische regels die specificeren waar een rubriek aan moet voldoen om als geldige invoer/uitvoer door het systeem geaccepteerd te worden. Deze regels beschrijven feitelijk het waardedomein voor de betreffende rubriek. Als voor de rubriek een waarde buiten dit domein aangeboden wordt, hoort het systeem op een gecontroleerde wijze – meestal met een foutmelding – de verwerking af te breken.

Syntactische regels kunnen in verschillende documenten vastgelegd zijn, maar meestal zijn ze beschreven in:

- De 'data dictionary' en andere gegevensmodellen waarin de kenmerken van alle gegevens gedetailleerd beschreven staan;
- Functionele specificaties van de betreffende functie of invoerscherm waarin de specifieke eisen aan de rubrieken beschreven staan.

De syntactische regels kunnen in willekeurige volgorde en onafhankelijk van elkaar getest worden. De dekkingsvorm die hierbij toegepast wordt, is:

- Afvinklijst.

In de praktijk wordt meestal gebruik gemaakt van de invoerschermen van gegevens om de syntactische controles te testen. Om praktische redenen wordt dit vaak gecombineerd met de zogenaamde *presentatietests* waarmee de opmaak van de schermen getest wordt. Overigens kunnen presentatietests zowel op invoer (schermen) als op uitvoer (lijsten; rapporten) toegepast worden. Opmaakregels kunnen in verschillende documenten vastgelegd zijn, maar meestal zijn ze beschreven in:

- Stijlgidsen die, vaak voor de hele organisatie, richtlijnen of regels bevatten voor zaken als kleurgebruik, lettertype/-grootte, schermopbouw, enzovoorts;
- Specificaties van de betreffende lijst of scherm waarin de opmaak concreet vastgelegd is.

Bij zowel de validatietest als de presentatietest wordt gebruik gemaakt van checklists waarin de controles beschreven staan die voor iedere rubriek of scherm toepasbaar zijn en getest kunnen worden.

Aandachtspunten bij de stappen

Ook voor de SYN worden in principe de generieke stappen (zie 14.4.1 "Inleiding") uitgevoerd. Echter de opbouw van een syntactische test is vrij triviaal: Iedere rubriekcontrole en opmaakcontrole wordt afzonderlijk getest. Iedere controle leidt tot één of meer testsituaties en iedere testsituatie leidt in het algemeen tot één testgeval.

Daarom beperkt deze paragraaf zich tot het toelichten van de eerste stap "identificeren testsituaties". Dit identificeren zal zich met name richten op het samenstellen van de relevante checklists en het op basis hiervan organiseren van de test.

1. Identificeren testsituaties

Bij de syntactische test kunnen twee soorten controles van toepassing zijn:

- Rubriekcontroles, voor de validatietest;
- Opmaakcontroles, voor de presentatietest.

Voor beide groepen controles is hieronder een overzicht gegeven van de meest voorkomende types controles. Deze kunnen gebruikt worden voor het samenstellen van een checklist die op iedere te testen rubriek/scherm toegepast wordt.

Overzicht van rubriekcontroles

- Datatype
  Bijvoorbeeld: numeriek; alfabetisch; alfanumeriek; enzovoorts.
- Veldlengte
  Vaak is de lengte van het invoerveld beperkt. Onderzoek wat er gebeurt als geprobeerd wordt om over deze lengte heen te gaan? (Hou de lettertoets eens een tijdje ingedrukt.)
- Invoer / Uitvoer
  Hier zijn 3 mogelijkheden:

I: Er wordt geen waarde getoond, maar deze moet of mag ingevoerd worden.
U: De waarde wordt getoond, maar mag niet gewijzigd worden.
UI: Er wordt een waarde getoond, maar die mag gewijzigd worden.

- Default
  Als de rubriek niet gevuld wordt, hoort het systeem de verwerking uit te voeren met de defaultwaarde.
  Als het een UI-veld (zie hierboven) betreft, dient de defaultwaarde getoond te worden.
- Verplicht / Niet verplicht
  Een verplichte rubriek mag niet leeg blijven.
  Een niet verplichte rubriek mag wel leeg blijven. In de verwerking wordt óf het gegeven leeg gelaten óf de defaultwaarde voor dit gegeven gebruikt.
- Selectiemechanisme
  Er moet een keuze gemaakt worden uit een aantal opgegeven mogelijkheden. Hierbij is het belangrijk of slechts één mogelijkheid gekozen mag worden of meerdere. Dit speelt vooral bij GUI's (Graphical User Interface), bijvoorbeeld bij
  – radio buttons (meerdere proberen te activeren);
  – check boxes (meerdere proberen te activeren);
  – drop down box (waarde proberen te wijzigen of leeg te maken).
- Domein
  Dit beschrijft alle geldige waarden voor deze rubriek. Het kan in principe op twee manieren weergegeven worden:
  – Opsomming
    Bijvoorbeeld {M, V, O};
  – Waardebereik
    Alle waarden tussen de opgegeven grenzen zijn toegestaan. Hierbij dienen met name de grenswaarden zelf getest te worden.
    Bijvoorbeeld [0, 100>, waarbij de symbolen aangeven dat het waardebereik van 0 tot 100 is, *in*clusief de waarde 0, maar *ex*clusief de waarde 100.

Tip

In de praktijk blijkt de waarde 0 (nul) nogal eens problemen te veroorzaken bij invoervelden. Het verdient aanbeveling om bij *ieder* invoerveld de waarde 0 uit te proberen.

- Speciale karakters
  Is het systeem bestand tegen speciale karakters zoals quotes, uitsluitend spaties, vraagtekens, Ctrl-karakters enzovoorts?
- Formaat

Voor sommige rubrieken zijn er specifieke eisen aan het formaat gesteld, bijvoorbeeld:
- Datum
  Bekende formaten zijn bijvoorbeeld YYYYMMDD of DD-MM-YY;
- Postcode
  Het formaat voor postcode is in principe per land verschillend. In Nederland is het formaat hiervoor “1111 AA” (vier cijfers gevolgd door een spatie en twee letters).

Overzicht van opmaakcontroles

- Kopregels / Voetregels
  Wordt er voldaan aan de standaards op dit gebied. Er kan bijvoorbeeld gedefinieerd zijn dat de volgende informatie aanwezig moet zijn:
  - Schermnaam of lijstnaam;
  - Systeemdatum of afdrukdatum;
  - Versienummer.
- Rubrieken
  Per rubriek zijn specifieke eisen aan opmaak gedefinieerd. Bijvoorbeeld:
  - Naamgeving van de rubriek;
  - Plaats van de rubriek op het scherm of overzicht;
  - Weergave van de rubriek zoals lettertype, kleur, enzovoorts.
- Overige schermobjecten (optioneel)
  Eventueel kunnen soortgelijke controles als bij “Rubrieken” toegepast worden op overige schermobjecten, zoals “push buttons” en “drop down lists”.

De checklist beschrijft in *algemene termen* de soort controle die uitgevoerd moet worden op een bepaalde rubriek of op een bepaald scherm. De *concrete eisen* aan de rubriek of opmaak die gerelateerd zijn aan deze soort controle, staan gedetailleerd beschreven in de betreffende systeemdocumentatie zoals schermbeschrijvingen en data dictionary.

De syntactische test is voldoende gespecificeerd als er een overzicht is van

- Alle rubrieken en schermen die getest moeten worden;
- Alle controles (checklist) die bij de betreffende rubriek/scherm getest moeten worden.

Tijdens de testuitvoering moet de tester de relevante systeemdocumentatie bij de hand houden voor de exacte details van de controles. Deze werkwijze heeft als bijkomend voordeel dat bij wijziging van de systeemdocumentatie de testspecificaties niet aangepast hoeven te worden.

Als alternatief kan er voor gekozen worden om de details uit de systeemdocumentatie over te nemen in de testspecificaties. Het voordeel hiervan is dat tijdens de testuitvoering niet meer in de systeemdocumentatie naar details gezocht hoeft te worden. Daarentegen heeft het als nadelen dat het onevenredig meer werk is om de test te specificeren en dat het gevaar bestaat dat de testspecificaties ‘uit de pas’ lopen met de systeemdocumentatie.

Als er sprake is van een groot aantal rubrieken en schermen, bestaat het gevaar dat het aantal uit te voeren controles ongewenst groot wordt. Daarnaast is de zwaarte van de fouten die over het algemeen worden gevonden met de syntactische test vrij laag. Daarom kan het zinvol zijn om de testinspanning te beperken door te prioriteren: Bepaal alle rubrieken/schermen die getest moeten worden en sorteer ze naar prioriteit;

- Bepaal alle controles die toegepast moeten worden op de rubrieken/schermen en sorteer ze naar prioriteit;
- Voer eerst de tests uit met de hoogste prioriteit controles en rubrieken/ schermen. Afhankelijk van het aantal en de ernst van de gevonden fouten kan besloten worden om al of niet door te gaan met tests van lagere prioriteit.

Een dergelijk mechanisme biedt tegelijk prima mogelijkheden om de test te managen en te rapporteren over voortgang, dekking en risico’s.

Uitgediept

In een matrix kan in één overzicht weergegeven worden welke controles op welke rubrieken gedaan moeten worden en welke prioriteit dit heeft. Iedere cel in de matrix is een testsituatie die getest moet worden. Met een “-” of arcering kan worden aangegeven dat de betreffende controle niet van toepassing is op die rubriek.

In tabel 14.18 is een voorbeeld gegeven voor de syntactische test voor de functie “Invoeren boeking”. De matrix kan eenvoudig uitgebreid worden voor ieder(e) functie/ scherm waarop een syntactische test gewenst is.

De prioriteiten zijn aangegeven met H(oog), M(iddel) en L(aag). In dit voorbeeld is de prioriteit op functieniveau bepaald en worden alle rubrieken van de geselecteerde functie getest. Er zou echter ook gekozen kunnen worden om verschillende prioriteiten aan de rubrieken binnen één functie toe te kennen. Zo’n verfijning is uiteraard een aanzienlijke hoeveelheid extra werk.

		Datatype	Formaat	Domein	I/U	Selectie	Verplicht	Veldlengte	Speciale karakters	Default
Invoeren boekingen	**H**	H	H	H	M	M	M	L	L	L
boeking-id			-			-				-
bestemming			-							-
maatschappij			-							-
reisdatum						-		-		
aantal passagiers			-			-				
klasse			-							
…	…									

Tabel 14.18: Matrix van rubrieken versus rubriekcontroles voor de SYN.

Tijdens testuitvoering kan dezelfde matrix gebruikt worden om weer te geven welke testsituaties gereed zijn en eventueel welke bevindingen daarbij gedaan zijn.

2. Opstellen logische testgevallen

De testsituaties uit stap 1 zijn direct de logische testgevallen.

3. Opstellen fysieke testgevallen

Indien gewenst kunnen de testsituaties verder uitgewerkt worden tot fysieke testgevallen en een testscript. Dit is met name zinvol als de test door onervaren testers uitgevoerd gaat worden.

Uitgediept

In tabel 14.19 is een voorbeeld gegeven van een syntactische test die fysiek uitgewerkt is. Hierin zijn 3 groepen testsituaties zichtbaar: de “altijd fout” situaties; situaties die soms fout en soms goed kunnen zijn; de “altijd goed” situaties.

Scherm 2.6	**Altijd Fout**				**F/G**	**F/G**	**Altijd Geldig**	
waarde / veld	**te veel tekens**	**te weinig tekens**	**verkeerde tekens**	**waarde buiten range**	**veld leeg laten**	**waarde nul**	**min. waarde**	**max. waarde**
bankgroep	> 5	< 5	¬ num	n.v.t.	FB	FB	00001	55555
vervdat	> 4	< 4	¬ num	¬ **01-**12	OK	FB	0001	9912
postcode	> 6	< 6	¬ alfanum	¬ 9999XX	FB	FB	1000AA	9999ZZ
pinind	> 1	< 1	¬ alfa	¬ {J,N}	FB	FB	J	N

Tabel 14.19: Voorbeeld van fysieke testgevallen bij de SYN.

De afkorting “FB” staat voor “foutboodschap”. Het symbool “¬” staat voor “niet”.

Meestal echter biedt de beschrijving van iedere testsituatie al voldoende informatie om de test te kunnen uitvoeren en is verdere fysieke uitwerking niet noodzakelijk. In dat geval wordt aanbevolen om tijdens de testuitvoering wel op enige wijze vast te leggen wat er concreet getest is, ten behoeve van de reproduceerbaarheid van de test. In de gevallen waarbij een fout gevonden wordt, is dit zelfs een voorwaarde.

Tip

De syntactische test is bij uitstek geschikt om te automatiseren. Naast het voordeel van reproduceerbaarheid, biedt de geautomatiseerde test vanzelf een gedetailleerde documentatie over wat er precies getest is en welke situaties

goed of fout gegaan zijn.

4. Vaststellen uitgangssituatie

Meestal zijn er geen speciale eisen aan de uitgangssituatie.

Uitzondering hierop is wellicht het testen van de syntax en layout van *uit*voer zoals lijsten en rapporten. Hierbij kan het voorkomen dat een bepaalde lijst met een bepaalde waarde in een bepaald veld pas geproduceerd kan worden na een complexe en langdurige serie handelingen.

Tip

Combineer het syntactisch testen van lijsten en rapporten zoveel mogelijk met de functionele tests (die de de benodigde variatie aan lijsten en rapporten produceren), of doe ze na afloop van dergelijke tests zodat de database reeds maximaal gevuld is met lijsten en rapporten.

14.4.12 Use case test (UCT)

Omschrijving

De use case test is een techniek die vooral wordt toegepast bij het testen van de kwaliteitsattributen inpasbaarheid, bruikbaarheid en gebruikersvriendelijkheid. De testbasis bevat minimaal de use cases en bij voorkeur ook het bijbehorende use case diagram. Er zijn diverse definities van het begrip use case in omloop. In deze paragraaf wordt de volgende definitie gebruikt:

Definitie

Een use case bevat een typische interactie tussen een gebruiker en een systeem. De use case beschrijft een compleet stuk functionaliteit dat een systeem aanbiedt aan een gebruiker en dat een voor de gebruiker observeerbaar resultaat oplevert.

Naast het bestaan van diverse use case definities, bestaan er ook diverse vormen van use case beschrijvingen. De vorm kan variëren van organisatie tot organisatie en zelfs van project tot project. De variaties hebben betrekking op het abstractieniveau, de scope en de mate van detail, waarin een use case is beschreven. Aangezien een use case op diverse manieren kan worden beschreven, is het verstandig om voor het toepassen van de use case test een controle uit te voeren om te onderzoeken of de gehanteerde use case beschrijving voldoende informatie bevat om voor de use case test te kunnen worden gebruikt. Deze controle kan het eenvoudigst met een checklist worden uitgevoerd (zie tip “checklist use cases”).

Tip

Checklist use cases

De detailinhoud van een checklist om te bepalen of een use case bruikbaar is voor het toepassen van de use case test, is afhankelijk van de manier waarop een use case is beschreven. Hier volgen enkele controles die als basis voor het maken van een ‘eigen’ checklist kunnen worden gebruikt:

- Is de (standaard voor project/organisatie) use case sjabloon volledig ingevuld?
- Is het use case diagram aanwezig?
- Is de use case een op zichzelf staande taak?
- Is het doel van de use case duidelijk?
- Is het duidelijk voor welke actoren de use case is bedoeld?
- Heeft de use case betrekking op de functionaliteit (dus niet op schermverloop)?
- Zijn alle voorziene alternatieve mogelijkheden beschreven?

- Zijn alle bekende uitzonderingen beschreven?
- Bevat de use case een volledig stappenplan?
- Is elke stap in de scenario('s) helder, ondubbelzinnig en compleet beschreven?
- Zijn alle in de use case genoemde actoren en stappen relevant voor het uitvoeren van de taak?
- Zijn de beschreven stappen uitvoerbaar?
- Is het resultaat van de stappen controleerbaar?
- Zijn de pre- en postcondities in overeenstemming met de use case?

De use case test richt zich op het afdekken van de interacties tussen de gebruiker en het systeem. De basistechniek die hierbij wordt gebruikt, is:

- Afvinklijst

Variaties op de use case test kunnen worden gecreëerd door het toepassen van andere basistechnieken, zoals:

- Paden
- Beslispunten: modified condition/decision coverage
- Pairwise testing

Bovengenoemde basistechniek "afvinklijst" is vrijwel altijd bruikbaar. De bruikbaarheid van de alternatieve basistechnieken is sterk afhankelijk van de inhoud van de use case beschrijvingen. In deze paragraaf wordt bij de uitwerking van een voorbeeld gebruik gemaakt van de basistechniek "afvinklijst". Voor een toelichting op de overige technieken wordt verwezen naar paragraaf 14.3 "Dekkingsvormen en basistechnieken".

Uitgediept

Use case diagram

Een use case beschrijft een (gedeelte van de) functionaliteit. Een use case diagram geeft de systeemgrenzen aan, geeft mogelijke onderlinge relaties tussen use cases weer, en laat vooral zien welke relaties er zijn tussen de actoren (gebruikers) en de use cases.

Een use case diagram is relatief eenvoudig. De drie belangijkste symbolen zijn:

- 'poppetje' om een actor aan te geven;
- ovaal om een use case aan te geven;
- lijnverbinding tussen actor en use case, of tussen use cases onderling (zie onder staande toelichting).

Use cases kunnen onderling twee soorten relaties hebben: "extend" of "include":

- Extend
  Een extend-relatie wordt gebruikt als een al aanwezige use case overeenkomt met een andere use case, maar iets extra's doet. Dit 'iets extra's' is uit de use case weggehaald en in een aparte use case geplaatst.
- Include
  Als een bepaald gedrag in meerdere use cases voorkomt, wordt dit er meestal uitgemodelleerd in plaats van herhaald in elke use case. Op deze wijze ontstaat een use case die gebruikt wordt (include-relatie) door andere use cases. Een simpel voorbeeld is een use case die de burgerlijke staat van een persoon nagaat. Deze kan worden gebruikt door zowel de use case "bepalen belastingtarief" als de use case "aanpassen burgerlijke staat". Deze wordt er dan vaak uitgemodelleerd als use case "bepalen burgerlijke staat".

De overeenkomst tussen extend en include is, dat in beide gevallen overeenkomstig gedrag verwijderd is om herhaling te voorkomen. Het verschil tussen beide is dat een include-relatie, in tegenstelling tot de extend-relatie, vaak geen actor meer kent. Daarbij wordt de include use case altijd en de extend use case optioneel uitgevoerd.

Voor meer informatie over use cases/models wordt verwezen naar de officiële Unified

Modeling Language (UML) documentatie van de Object Management Group (www.omg.org).

Aandachtspunten bij de stappen

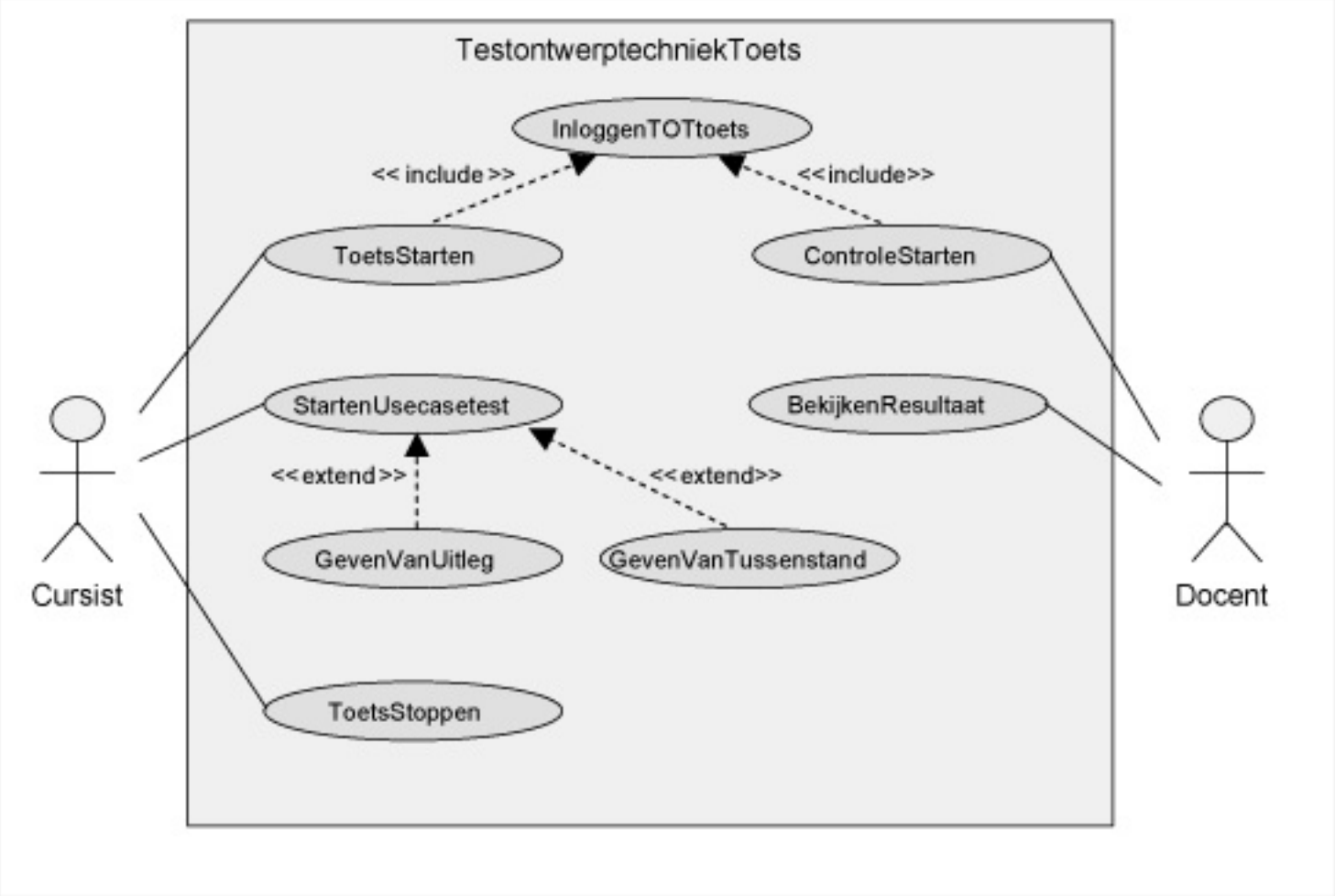
In deze paragraaf wordt de use case test stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie 14.4.1 “Inleiding”) als uitgangspunt genomen. Tevens is een voorbeeld uitgewerkt dat bij iedere stap laat zien hoe deze techniek in zijn werk gaat. In het voorbeeld is volgens bovenstaande beschrijving een use case diagram opgesteld. Aangezien er geen uniforme afspraken bestaan over hoe een beschrijving van een use case er moet uitzien, zijn in het voorbeeld alleen de voor het voorbeeld relevante use case componenten opgenomen.

Voorbeeld

Figuur 14.19 toont een voorbeeld van een use case diagram waarin de cursist (“actor”) een applicatie (“TestontwerptechniekToets”) kan starten. Na de inlogprocedure doorlopen te hebben kiest de cursist voor een bepaalde testontwerptechniek waarop hij wil worden getoetst. In het voorbeeld is alleen de use case “StartenUsecasetest” als mogelijk te toetsen testontwerptechniek beschreven.

Tijdens de toets is de mogelijkheid aanwezig dat de cursist bij het geven van een fout antwoord een uitleg krijgt. Ook het geven van een tussenstand met betrekking tot het aantal gegeven goede antwoorden, behoort tot de mogelijkheid. Nadat er een bepaald aantal vragen is gesteld, stopt de applicatie.

De docent (“actor”) kan de vorderingen en de resultaten van de cursist volgen.



Figuur 14.19: Use case diagram “TestontwerptechniekToets”.

Als voorbeeld worden enkele use case beschrijvingen gegeven:

Name	**ToetsStarten**
Actor	Cursist
Preconditions	Cursist heeft opleiding testontwerptechnieken gevolgd
Primary scenario	1. De cursist start het “TestontwerptechniekToets” programma 2. Include <InloggenTOTtoets> 3. De cursist maakt een keuze uit de testontwerptechnieken 4. De cursist kiest eventueel voor de optie “geven van uitleg” 5. De cursist kiest eventueel voor de optie “geven van tussenstanden” 6. Het programma staat klaar voor de eerste vraag

Postconditions	Cursist kan beginnen met de uitgekozen test

Name	**StartenUsecasetest**
Actor	Cursist
Preconditions	Cursist heeft opleiding testontwerptechnieken gevolgd
Primary scenario	1. De use case start als de cursist voor de eerste keer op de vraagknop drukt 2. Zolang er minder dan tien vragen zijn gesteld 2.1 De computer genereert een vraag met betrekking tot de use case test 2.2 De cursist leest de vraag, bedenkt het antwoord en tikt dit in 2.3 Na controle drukt de cursist op “enter” 2.4 De computer leest het antwoord 2.5 Als het antwoord verwerkbaar is dan 2.5.1 wordt de beoordeling “goed” gegeven als het antwoord overeenkomt met dat van de computer 2.5.2 wordt de beoordeling “fout” gegeven als het antwoord niet overeenkomt met dat van de computer 2.6 Anders krijg de cursist de melding: “antwoord is niet verwerkbaar en wordt daarom fout gerekend” <GevenVanTussenstand> 2.7 De cursist denkt na over een fout antwoord <GevenVanUitleg> 2.8 De cursist drukt op de vraagknop 3. De cursist krijgt een beoordeling in de vorm van een cijfer na tien vragen 4. Het programma stopt
Exceptions	1. Cursist drukt op de vraagknop, terwijl de vorige vraag niet is beantwoord 2. Cursist tikt iets in het invoerveld, terwijl de vraag al is beantwoord 3. Cursist stopt, terwijl er nog geen tien vragen zijn beantwoord
Postconditions	De cursist krijgt een cijfer voor de toets

Name	**GevenVanTussenstand**
Actor	Cursist
Preconditions	Cursist heeft bij het starten van de toets voor het geven van de tussenstand gekozen
Primary scenario	• Het goede antwoord verhoogt het aantal goede antwoorden met één. Op grond van dit aantal wordt een tussenstand gegenereerd • De tussenstand wordt in een statusregel weergegeven

Uitgediept

Use case componenten die naast de bovengenoemde componenten (name, actor, preconditions, primary scenario, exceptions en postconditions) vaak in een use case sjabloon worden gebruikt, zijn:

- Scope (betreft het bijvoorbeeld een systeem of subsysteem)
- Level (betreft het bijvoorbeeld een primaire taak of subfunctie)
- Stakeholder (betreft de belanghebbende(n): bijvoorbeeld cursist, docent, werkgever en/of klant)
- Trigger (het ‘event’ dat ervoor zorgt dat de use case start, wordt ook wel weergegeven als de eerste stap in het use case scenario)
- Priority (mate van belangrijkheid van de use case)
- Response time (de beschikbare tijd voor het uitvoeren van de use case)
- Frequency (aantal keren dat de use case wordt uitgevoerd)
- Secondary actors (andere bij de use case betrokken actors)

Het gebruik van bovenstaande use case componenten is niet dwingend. Afhankelijk van de aard van de use case worden componenten toegevoegd of weggelaten. Uiteraard zijn basiscomponenten als “Name”, “Actor”, “Preconditions”, “Primary scenario” en “Postconditions” vrijwel altijd in een use case beschrijving aanwezig.

Hieronder is per stap uitgewerkt hoe de use case test wordt toegepast op dit voorbeeld:

1. Identificeren testsituaties

Het afleiden van testsituaties uit de use case is in grote mate afhankelijk van het detailniveau waarop de use case is

beschreven. Bij weinig detail kan het zijn dat voor een use case slechts één testsituatie (bijvoorbeeld het doel van de use case) kan worden beschreven. Voor deze testsituaties kunnen dan op dat moment geen logische testgevallen worden gemaakt, omdat daarvoor te weinig informatie aanwezig is. Bevat de use case meer detail, dan kunnen er uiteraard gedetailleerde testsituaties worden onderscheiden, die direct als de logische testgevallen worden beschouwd. In alle gevallen worden de onderkende testsituaties op een afvinklijst opgenomen, zodat in het vervolgtraject kan worden afgevinkt of er voor elke testsituatie minimaal één logisch testgeval is gemaakt.

Aangezien er geen uniforme beschrijving voor een use case bestaat, is het ook niet mogelijk een formele manier voor het afleiden van testsituaties te geven. Afhankelijk van de kennis en kunde van de tester zal dat de een wat makkelijker afgaan dan de ander (zie ook tip).

Tip

Afhankelijk van de wijze waarop een use case is beschreven, kan het uitvoeren van de volgende stappen helpen de gedachten te ordenen bij het identificeren en beschrijven van testsituaties:

1. Zoek variabelen die een reactie van het systeem of de omgeving tot gevolg hebben.
   Voorbeelden van variabelen zijn: invoergegevens, uitvoergegevens, omgevingvariabelen die de actor tot een bepaald gedrag dwingen, status van het systeem, enzovoort.
2. Bepaal het domein van de variabelen.
3. Bepaal welke variabelen een relatie met elkaar hebben.
4. Combineer gerelateerde variabelen tot een testsituatie en beschrijf deze. Een testsituatie bevat minimaal de relatie tussen de variabelen, bepaalde waarde(n) uit het domein, de startsituatie en een speci fiek beschreven resultaat.

Het resultaat van de eerste drie stappen kan er voor de use case “StartenUse casetest” als volgt uit zien:

Variabele	Domein	Relatie met
Tussenstand geven	{J, N}	Gegeven antwoord
Uitleg geven	{J, N}	Gegeven antwoord
Gegeven antwoord	{Goed, Fout}	Verwerkbaar antwoord, Tussenstand geven, Uitleg geven
Verwerkbaar antwoord	{J, N}	Gegeven antwoord
Aantal gegeven antwoorden	{0 - 10}	Programma

Met behulp van bovenstaande tabel en de use case beschrijving kunnen testsituaties geïdentificeerd en beschreven worden (stap 4).

Voorbeeld uitwerking

Stel dat de use cases uit het voorbeeld “TestontwerptechniekToets” vrijwel geen details zouden bevatten, dan zou een afvinklijst met testsituaties (dit zijn *geen* logische testgevallen) er als volgt uit kunnen zien:

Use case “Toetsstarten”

1a De cursist moet met ‘eigen’ settings op de toetsapplicatie kunnen aanloggen.
1b De cursist moet een toets voor een testontwerptechniek kunnen kiezen.

Use case “StartenUsecasetest”

2a De cursist moet de toets “use case test” kunnen afleggen.
2b De cursist moet optioneel uitleg bij een fout antwoord kunnen krijgen.
2c De cursist moet optioneel een tussenstand kunnen krijgen.

Use case “ControleStarten”

3. De docent moet met ‘eigen’ settings op de toets(controle)applicatie kunnen aanloggen.

Use case “BekijkenResultaat”

4. De docent moet de vorderingen en resultaten van een cursist kunnen volgen.

Het voorbeeld bevat echter meer details, waardoor de testsituatiesbeschrijving niet tot een afvinklijst beperkt hoeft te blijven. In onderstaande tabel zijn als voorbeeld enkele testsituaties voor de use case “StartenUsecasetest” beschreven. Deze testsituaties zijn *wel* direct de logische testgevallen. Voor de beschrijving van een testsituatie is een op de use case beschrijving lijkende lay-out gekozen.

Use Case Name	**StartenUsecasetest**
Test case ID	**1**
Test case purpose	Controleer of de computer bij de eerste keer drukken op de vraagknop een vraag genereert
Priority	Middel
Actor	Cursist
Precondition	Kies bij het starten van de toets voor “Usecasetest”
Trigger	Druk op vraagknop
Postconditions	De eerste vraag over de “Usecasetest” wordt getoond
Test case ID	**2**
Test case purpose	Controleer of er een tussenstand wordt gegeven bij een goed antwoord
Priority	Laag
Actor	Cursist
Precondition	Kies bij het starten van de toets voor de optie “Tussenstand geven”. Het gegeven antwoord is verwerkbaar
Trigger	Tik goed antwoord in en druk op “enter”
	De melding “goed” wordt getoond en er wordt een tussenstand getoond
Test case ID	**3**
Test case purpose	Controleer of er geen uitleg wordt gegeven bij een fout antwoord
Priority	Hoog
Actor	Cursist
Precondition	Kies bij het starten van de toets niet voor de optie “Uitleg geven” Het gegeven antwoord is verwerkbaar.
Trigger	Tik fout antwoord in en druk op “enter’
Postcondition	De melding “fout” wordt getoond en er wordt geen uitleg gegeven
Test case ID	**4**
Test case purpose	Controleer of het programma stopt als er na 10 gegeven antwoorden op de vraagknop wordt gedrukt
Priority	Middel
Actor	Cursist
Precondition	Er zijn 10 antwoorden gegeven
Trigger	Druk op vraagknop
Postcondition	Er wordt een eindcijfer getoond en het programma stopt

De component “Priority” kan worden gebruikt voor het aangeven of het testgeval altijd of optioneel mag worden uitgevoerd. Verder kan de component worden gebruikt om de volgorde van uitvoering te bepalen.

2. Opstellen logische testgevallen

Als stap “1-Identificeren testsituaties” in een afvinklijst heeft geresulteerd, kunnen er (op dit moment) op basis van de use cases géén logische- en fysieke testgevallen worden opgesteld. In dat geval moet in andere delen van de testbasis worden gezocht naar aanvullende informatie om alsnog de testgevallen op te kunnen stellen. Heeft stap 1 wel gedetailleerde testsituaties opgeleverd, dan zijn dit direct de logische testgevallen.

In een UCT case traceability matrix wordt bijgehouden (door afvinken) of uiteindelijk voor alle onderkende testsituaties minimaal één logisch testgeval is gemaakt.

Voorbeeld uitwerking

Bij het afvinklijstvoorbeeld kan op een gegeven moment, op basis van informatie uit andere delen van de testbasis, toch worden gestart met het maken van logische testgevallen. De (hier met fictieve waarden ingevulde) UCT test case traceabilty matrix zou er dan als volgt uit kunnen zien:

	Logisch testgeval								
Testsituatie	**TG-1**	**TG-..**	**TG-..**	**TG-..**	**TG-..**	**TG-..**	**TG-..**	**TG-..**	**TG-n**
1a	√		√						
1b							√		
2a									
2b				√					
2c								√	

3									
4		√			√				

Uit de matrix valt af te lezen dat er nog geen logische testgevallen (TG) zijn gemaakt voor testsituaties 2a en 3. Verder valt af te lezen dat bijvoorbeeld testsituatie 1a in diverse logische testgevallen voorkomt.

De triviale UCT test case traceabilty matrix voor het voorbeeld met de gedetailleerde testsituaties (die tevens de logische testgevallen zijn) ziet er als volgt uit:

	Logisch testgeval			
Testsituatie	**TG-1**	**TG-2**	**TG-3**	**TG-4**
1	√			
2		√		
3			√	
4				√

3. Opstellen fysieke testgevallen

Geen bijzonderheden.

4. Vaststellen uitgangssituatie

Geen bijzonderheden.

Noot

1 Voor het uiteindelijke aantal combinaties maakt dit niet uit. Reken maar na: in het verzendkostenvoorbeeld wordt één tabel gemaakt met in totaal vier condities. Dit leidt tot één tabel met maximaal 24 = 16 combinaties. In het voorbeeld zou de tabel in twee tabellen (“berekening standaardverzendkosten” en “berekening afstandstoeslag”) kunnen worden gesplitst. Beide tabellen zouden dan uit 22 = 4 testkolommen bestaan. Die in combinatie met elkaar 4*4, eveneens 16 combinaties zouden opleveren. Omdat wel of niet splitsen voor het uiteindelijke resultaat niet uitmaakt, is het splitsen van tabellen met meer dan vijf condities in meerdere tabellen aan te bevelen, omdat hierdoor de afzonderlijke tabellen leesbaar en overzichtelijk blijven.

15 Toetstechnieken

15.1 Inleiding

Het resultaat van een toetsing is in belangrijke mate afhankelijk van de houding van de toetser(s). Bij toetsen worden immers vaak documenten beoordeeld die door iemand anders zijn gemaakt. Dit resulteert dan meestal in een lijst van bevindingen die bestemd zijn voor de auteur van het desbetreffende document. Afhankelijk van hoe de bevindingen zijn genotuleerd of gecommuniceerd, kan het gebeuren dat de auteur zich 'aangevallen' voelt. De kans is groot dat hierdoor een negatieve sfeer rond het toetsproces komt te hangen. Het is daarom belangrijk te beseffen dat het niet de intentie van de auteur is geweest om zaken bewust 'verkeerd' op te schrijven. Daarbij is het tevens van belang om te realiseren dat het toetsproces als uiteindelijk doel heeft om met zijn állen een zo goed mogelijk eindproduct op te leveren. Toetstechnieken zijn immers zeer geschikt om kwaliteitsverbetering van producten te realiseren. Dit geldt niet alleen voor de beoordeelde producten zelf, maar zeker ook voor andere producten. De bevindingen uit het toetsproces kunnen voor bijvoorbeeld het ontwikkelproces aanleiding zijn om procesverbeteringen aan te brengen.

Uit onderzoek blijkt echter dat toetsprocessen, ondanks hun bewezen waarde, niet altijd soepel worden geïmplementeerd of uitgevoerd. Enkele oorzaken die uit een enquête naar voren kwamen [Hendriks, 1999]:

- voor 56% van de auteurs was het moeilijk om zich los te koppelen van hun werk
- 48% van de toetsers had niet de nodige training gekregen
- 47% van de auteurs was bang dat de gegevens tegen hen gebruikt zouden worden.

In dit hoofdstuk wordt, na een algemene toelichting op toetsen, een drietal technieken nader uitgelegd: inspecties, reviews en walkthroughs. De laatste paragraaf van het hoofdstuk bevat een keuzematrix toetstechnieken, die gebruikt kan worden bij het kiezen van een techniek.

Als basis voor dit hoofdstuk is de "IEEE Std 1028-1997 Standard for Software Reviews" [IEEE, 1998] gebruikt, die is aangevuld met praktijkervaringen.

Tip

Toetsen is één van de aandachtsgebieden in het 'test process improvement' (TPI) model [Koomen, 2006]. Voor het implementeren én verbeteren van het toetsproces bevat het TPI boek enkele suggesties (paragraaf 7.19 "Toetsen").

15.2 Toetsen toegelicht

Tussenproducten

Tijdens het systeemontwikkelproces worden diverse tussenproducten ontwikkeld. Deze hebben afhankelijk van de gekozen methode een bepaalde vorm, inhoud en een relatie met elkaar en kunnen daarop worden getoetst (zie ook paragraaf 2.3.3 "Test- en systeemontwikkelproces").

Definitie

Toetsen is het beoordelen van de tussenproducten in het systeemontwikkelproces.

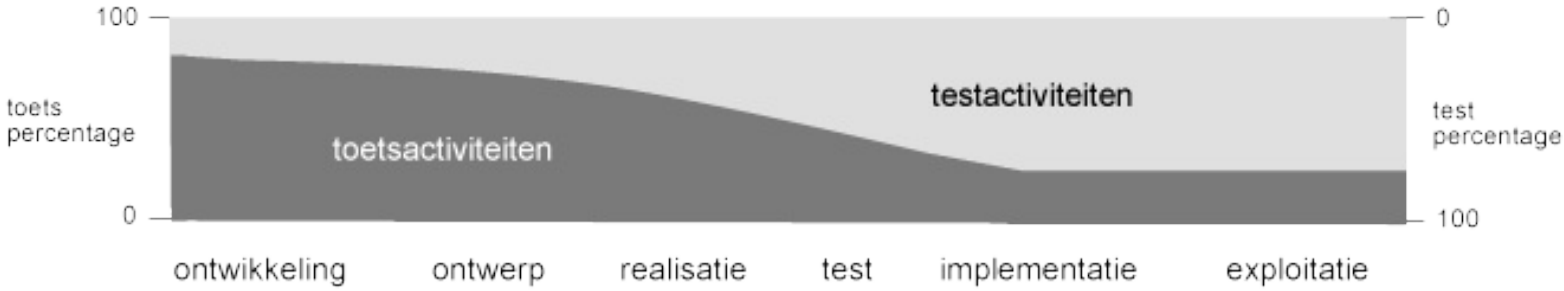
Hierbij hoeft het toetsen op tussenproducten zeker niet beperkt te blijven tot de ontwikkeldocumenten. Er kan op alle niveaus van documentatie worden getoetst:

- functioneel-/technisch ontwerp
- requirementsdocument
- managementplan
- ontwikkelplan
- testplan

- onderhoudsdocumentatie
- gebruikers-/installatie handleiding
- software
- release note
- testontwerp
- testscript
- prototype
- screen print
- enzovoort.

Plaats van het toetsen

Naast de eerder beschreven kwaliteitsverbetering is een ander groot belang van toetsen dat de fouten veel eerder in het (ontwikkel)proces kunnen worden gevonden dan bij testen (zie figuur 15.1). Bij toetsen worden immers tussenproducten beoordeeld en niet, zoals bij testen, de eindproducten. En hoe eerder een fout wordt gevonden, hoe eenvoudiger én goedkoper een fout kan worden hersteld [Boehm, 1981].



Figuur 15.1: Toetsen en testen versus systeemontwikkelfasering

Van toetsen is keer op keer bewezen dat dit de meest efficiënte en effectieve manier is om fouten uit de tussenproducten van een systeem te halen (zie de praktijkvoorbeelden in paragraaf 6.5 “Fase Voorbereiding”). Daarbij is toetsen een eenvoudig in te richten proces, omdat er bijvoorbeeld geen software ‘gerund’ en geen omgeving ingericht hoeft te worden. Voldoende redenen dus om een toetsproces in te richten.

In de praktijk verzanden de goede bedoelingen echter nog wel eens in praktische uitvoeringsproblemen: “Kun je even naar deze zes mappen met de functionele specificaties kijken? Overmorgen graag weer terug want dan begint de bouw.”

Formele toetstechnieken, in de vorm van procesbeschrijvingen en checklists, helpen dit beheersbaar te maken. Een formele toetstechniek wordt gekenmerkt door het feit dat het toetsen door meer personen in teamverband wordt uitgevoerd, bevindingen worden gedocumenteerd (zie tip) en er beschreven procedures voor het uitvoeren van de toetsactiviteiten bestaan.

Tip

De toetsers doen vele bevindingen en vaak wordt er voor gekozen om deze in een bevindingenadministratie op te nemen. Na afloop van de toetsbijeenkomst moeten dan bijvoorbeeld rubrieken als status, ernst en actie van de bevindingen worden geactualiseerd. Dit zijn arbeidsintensieve en tijdrovende activiteiten en kunnen als volgt tot een minimum worden beperkt:

- Toetsers leggen hun opmerkingen vast in een toetsformulier.
- Tijdens de toetsbijeenkomst worden deze opmerkingen besproken.
- Na afloop van de bijeenkomst worden alleen de belangrijkste opmerkingen als bevinding in een bevindingenadministratie geregistreerd. Voor meer informatie over de bevindingenprocedure wordt verwezen naar hoofdstuk 12 “Bevindingenbeheer”.

Een toetsformulier bevat de volgende aspecten:

Per formulier

- identificatie van toetser

  - naam
  - rol
- identificatie van het tussenproduct
  - documentnaam
  - versienummer-/datum
- toetsprocesgegevens
  - aantal getoetste pagina's
  - bestede toetstijd
- algemene indruk van het tussenproduct

Per opmerking

- uniek volgnummer
- duidelijke verwijzing naar de plaats in het tussenproduct waar de opmerking betrekking op heeft (bijvoorbeeld door het vermelden van hoofdstuk-, paragraaf-, blad-, regel-, requirementsnummer)
- beschrijving van de opmerking
- belang van de opmerking (bijvoorbeeld hoog, middel, laag)
- vervolgacties (door *auteur* in te vullen met bijvoorbeeld voldaan, gedeeltelijk voldaan, niet voldaan)

Noot: Bovenstaand formulier is vooral bruikbaar bij het reviewen van documenten. Voor het reviewen van software of bij een walkthrough van een prototype moeten de aspecten op het formulier enigszins worden aangepast.

Er bestaan diverse zwaartes van toetstechnieken. Dit is van belang omdat niet elk tussenproduct even zwaar hoeft te worden getoetst. Daarom wordt er vaak in het mastertestplan een toetsstrategie opgenomen. Evenals een teststrategie is een toetsstrategie van groot belang om de inspanning optimaal in te zetten én als communicatiemiddel naar de opdrachtgever toe. Met een strategiebepaling wordt geanalyseerd wat, waar en hoeveel moet worden getoetst om de optimale balans te vinden tussen de gewenste kwaliteit en de hoeveelheid tijd c.q. geld die daarvoor nodig is. Er vindt optimalisatie plaats met het doel de beschikbare resources op de juiste wijze te verdelen over de uit te voeren activiteiten.

Nadere indeling toetstechnieken

Na enkele algemene tips worden in de volgende paragrafen de technieken volgens onderstaande indeling beschreven:

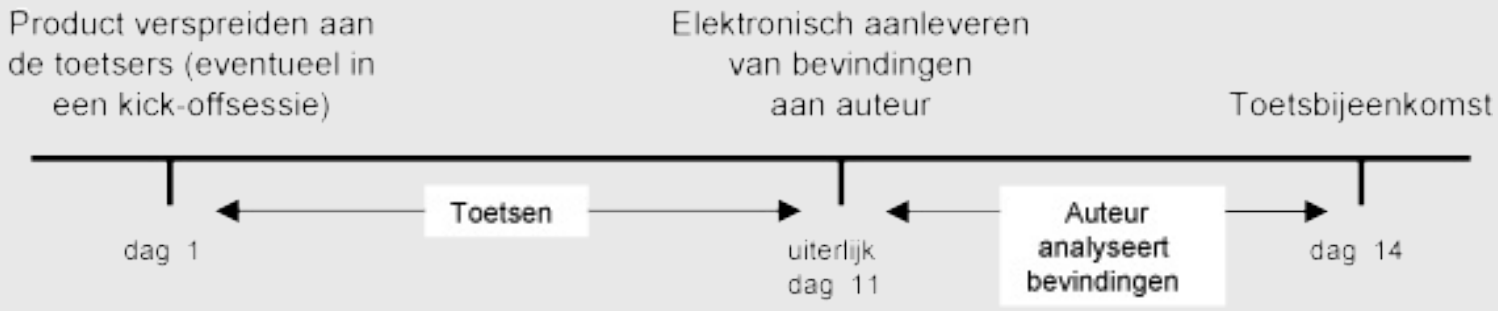
- Inleiding
  Beschrijving van het doel van de techniek en de producten waarop de techniek kan worden toegepast.
- Verantwoordelijkheden
  Beschrijving van welke rollen aan deelnemers worden toegekend.
- Entrycriteria
  Beschrijving van de noodzakelijke producten en voorwaarden waaraan voldaan moet zijn voordat het toetsen kan worden gestart.
- Procedures
  Beschrijving van:
  - hoe een toets te organiseren (plannen)
  - de te volgen werkwijze
  - de gewenste voorbereiding door de deelnemers
  - hoe toetsresultaten worden besproken en vastgelegd
  - hoe herstelwerkzaamheden worden uitgevoerd en gecontroleerd.
- Exitcriteria
  Beschrijving van de op te leveren documenten en de voorwaarden waaronder het product het toetsproces kan verlaten.

Tips

Bij het gebruik van een toetstechniek blijken er in de praktijk enkele kritieke succesfactoren aan te wijzen:

- De auteur moet zijn vrijgemaakt van andere werkzaamheden om aan het toetsproces deel te kunnen nemen en om de resultaten te kunnen verwerken.
- Auteurs mogen niet worden afgerekend op de toetsresultaten.
- Toetsers moeten een (korte) opleiding in de specifieke toetstechniek hebben gehad.

- Er moet genoeg (voorbereidings)tijd beschikbaar zijn tussen het aanbieden van de producten aan de toetsers en de toetsbijeenkomst (bijvoorbeeld twee weken). Eventueel kunnen de producten tijdens een kick-off beschikbaar worden gesteld. Zie figuur 15.2 voor een mogelijke planning van het toetstraject.
- De notulist moet ervaren en goed geïnstrueerd zijn. Het notuleren van alle bevindingen, acties, beslissingen en aanbevelingen van de toetsers is van doorslaggevend belang voor het laten slagen van een inspectieproces. Soms neemt de auteur de rol van notulist op zich, maar dit heeft als nadeel dat de auteur daardoor de kans loopt delen van de discussie te missen (aandacht verdelen over schrijven en luisteren).
- De omvang van het te toetsen tussenproduct en de beschikbare voorbereidings- en bijeenkomsttijd moeten op elkaar afgestemd zijn.
- Maak heldere vervolgafspraken. Spreek af wanneer en in welke versie van het tussenproduct de afgesproken wijzigingen moeten zijn doorgevoerd.
- Het geven van feedback door de auteur aan de toetsers (waardering voor de door hen geleverde bijdrage).



Figuur 15.2: Mogelijke planning toetstraject

In de planning (figuur 15.2) is geen rekening gehouden met eventueel uit te voeren activiteiten ná de toetsbijeenkomst. Dit kunnen activiteiten zijn als: aanpassen van het product, hertoetsen en het definitief accepteren van het product. Deze activiteiten moeten, indien relevant, nog aan de planning worden toegevoegd. Hierbij kunnen de activiteiten aanpassen van het product en hertoetsen een iteratief karakter hebben.

Praktijkvoorbeeld

In een organisatie waar de time-to-market van diverse aanpassingen aan het informatiesysteem kort moest zijn, werden deze aanpassingen door middel van een groot aantal kortlopende incrementen geïmplementeerd.

Van de testers werd verwacht dat zij de ontwerpen (in de vorm van use cases) reviewden. Hierbij was een doorlooptijd van twee weken voor een reviewtraject geen reële optie. Als oplossing werd gekozen voor de maandag als vaste reviewdag. De ‘spelregels’ waren:

- de ontwerpers leverden voor 9:00 één of meer te reviewen documenten bij de testmanager aan (werden er geen documenten aangeleverd, dan werd er op deze maandag niet gereviewed)
- de omvang van alle documenten samen mocht het totaal van 30 A4-tjes niet te boven gaan
- de testmanager bepaalde welke tester wat moest reviewen
- voor 12:00 moesten de reviewopmerkingen van de testers in het bezit van de betreffende ontwerper zijn
- de reviewbijeenkomst werd van 14:00 uur tot 15:00 gehouden
- streven was dat de ontwerper het document nog dezelfde dag zou aanpassen (afhankelijk van de zwaarte van de opmerkingen, werd dit soms een dag later).

15.3 Inspecties

Er bestaan diverse invullingen van het inspectieproces. In deze paragraaf wordt een algemene invulling gegeven. Voor specifieke invullingen wordt verwe zen naar [Gilb, 1993] en [Fagan, 1986].

Inleiding

Bij het uitvoeren van een inspectie wordt een formele werkwijze gevolgd, waarbij producten nauwkeurig worden gelezen

door een groep deskundigen. Dit kan elk van de in paragraaf 15.2 genoemde documenten zijn. Bij deze producten, die 100% gereed moeten zijn, wordt gelet op afwijkingen van vooraf afgesproken criteria. Naast het toetsen of de oplossing goed is verwerkt, is een inspectie vooral gericht op het verkrijgen van consensus over de kwaliteit van een product. Er moet worden gecontroleerd op bijvoorbeeld het voldoen van het product aan bepaalde standaards, specificaties, regelgeving, richtlijnen, plannen en/of procedures. Vaak worden de criteria, waarop moet worden geïnspecteerd, verzameld en vastgelegd in een checklist. Het doel van de inspectie is om de auteur te helpen bij het vinden van zoveel mogelijk afwijkingen in de daarvoor beschikbaar gestelde tijd.

Verantwoordelijkheden

Bij het uitvoeren van een inspectie zijn drie tot zes deelnemers betrokken.

Mogelijke rollen zijn:

- Moderator
  De moderator bereidt het inspectieproces voor en leidt dit proces. Dit houdt in het plannen van het inspectieproces, het maken van afspraken, het bepalen van product- en groepomvang en het verantwoordelijk zijn voor het vastleggen van alle gedane bevindingen gedurende het inspectieproces.
- Auteur
  De auteur vraagt een inspectie aan. Tijdens het inspectieproces licht de auteur onduidelijkheden in het product toe. Verder moet de auteur zeker stellen dat hij de bevindingen van de inspecteurs begrijpt.
- Notulist
  De notulist legt tijdens het inspectieproces alle bevindingen, acties, beslissingen en aanbevelingen van de inspecteurs vast.
- Inspecteur
  De inspecteur probeert voorafgaand aan de inspectiebijeenkomst zoveel mogelijk bevindingen te vinden en te documenteren. Vaak wordt een inspecteur hierbij, door de moderator, gevraagd dit vanuit één bepaald gezichtspunt te doen. Bijvoorbeeld vanuit projectmanagement-, test-, ontwikkel-, of kwaliteitsoogpunt (zie ook kader ‘perspective based reading’). Ook kan een inspecteur worden gevraagd het gehele product te inspecteren op correct gebruik van een bepaalde standaard.

Op de auteur na kunnen alle deelnemers één of meer van bovenstaande rollen invullen, waarbij alle deelnemers (met uitzondering van de auteur) in ieder geval de rol van inspecteur hebben. Hierbij mag geen van de deelnemers de leidinggevende zijn van één van de andere deelnemers, omdat anders de kans bestaat dat deelnemers terughoudend zijn bij het geven van hun bevindingen.

Uitgediept

Perspective based reading

Deelnemers aan een toetsactiviteit beoordelen een product vaak met hetzelfde doel en vanuit hetzelfde gezichtspunt. Hiernaast ontbreekt meestal een systematische aanpak bij de voorbereiding. Daardoor is de kans groot dat door de diverse deelnemers dezelfde soort fouten worden gevonden. Het gebruik van een goede leestechniek kan hier verbetering in aanbrengen. Een veelgebruikte techniek is de perspective based reading (PBR) techniek. Kenmerken van PBR zijn:

- Deelnemers toetsen een product vanuit één bepaald gezichtspunt (bijvoorbeeld als ontwikkelaar, tester, gebruiker, projectmanager).
- Benadering vanuit de *wat* en *hoe* vragen. In een procedure is vastgelegd wát de productonderdelen zijn die vanuit één bepaald gezichtspunt moeten worden getoetst en hóe deze moeten worden getoetst. Vaak wordt een procedure (scenario) per gezichtspunt (perspective) opgesteld.

PBR behoort tot de groep van ‘scenario based reading’ (SBR) technieken. Andere SBR technieken die door de deelnemers aan toetsactiviteiten kunnen worden gebruikt, zijn defect reading, scope reading, use based reading en horizontal/vertical reading.

Voor meer informatie wordt verwezen naar bijvoorbeeld [Laitenberger, 1995] en [Basili, 1997].

Entrycriteria

Het inspectieproces kan worden gestart als:

- Het doel van de inspectie en de inspectieprocedure duidelijk zijn.
- Het product 100% gereed (maar nog niet definitief) is, én van spellingsfouten is ontdaan, aan afgesproken standaards voldoet, begeleidende documentatie beschikbaar is, verwijzingen naar referenties juist zijn, enzovoort.
- De bij de inspectie te gebruiken checklists en inspectieformulieren voor het vastleggen van bevindingen aanwezig zijn.
- (eventueel) Een lijst met al bekende aanwezige bevindingen beschikbaar is.

Procedures

Het inspectieproces kan worden verdeeld in de fasen planning, kick-off, voorbereiding en uitvoering:

Planning

Wanneer het te inspecteren product aan de entrycriteria voldoet, wordt door de moderator een inspectiebijeenkomst georganiseerd. Dit houdt onder andere in: datum en plaats voor de bijeenkomst bepalen, team vormen, rollen toekennen, (door auteur aangeleverde) te inspecteren producten onder de deelnemers verspreiden en afspraken maken over de termijn waarop de auteur over de bevindingen moet kunnen beschikken.

Kick-off

De kick-off bijeenkomst wordt gehouden voordat de eigenlijke inspectie plaatsvindt. Deze bijeenkomst is optioneel en wordt door de moderator georganiseerd om de volgende redenen:

- Als er deelnemers uitgenodigd zijn, die nog niet eerder deelgenomen hebben aan een inspectie, geeft de moderator een korte inleiding in de techniek en de werkwijze die daarbij wordt gehanteerd.
- De auteur van het te inspecteren product geeft een inleiding op het product.
- Als er verbeteringen of wijzigingen te melden zijn in de te volgen werkwijze tijdens de inspectie, wordt hierop een korte toelichting gegeven.

Indien een kick-off wordt gehouden, worden tijdens deze bijeenkomst de documenten uitgereikt en kunnen de rollen worden toegelicht.

Voorbereiding

Voor het houden van een zo effectief en efficiënt mogelijke inspectiebijeenkomst, is een goede voorbereiding noodzakelijk. Tijdens de voorbereiding zoeken de inspecteurs naar bevindingen in de te inspecteren producten en leggen deze vast in het inspectieformulier. De moderator verzamelt deze formulieren, classificeert de bevindingen en stelt het resultaat hiervan tijdig aan de auteur beschikbaar, zodat de auteur zich kan voorbereiden.

Uitvoering

Het doel van de inspectiebijeenkomst is niet alleen het vastleggen (overnemen) van de door de deelnemers, tijdens de voorbereiding, geconstateerde bevindingen. Ook het doen van nieuwe bevindingen tijdens de bijeenkomst en het impliciet uitwisselen van kennis zijn belangrijke doelstellingen. Tijdens de bijeenkomst zorgt de moderator ervoor dat pagina voor pagina de bevindingen op een efficiënte wijze worden geïnventariseerd. De bevindingen worden door de notulist vastgelegd in een bevindingenlijst. Cosmetische fouten worden veelal niet geregistreerd, maar na afloop van de bijeenkomst aan de auteur overhandigd.

Aan het eind van de bijeenkomst neemt de moderator met alle deelnemers de door de notulist opgestelde bevindingenlijst door om er zeker van te zijn dat deze lijst compleet is en de bevindingen accuraat zijn vastgelegd. Onjuistheden worden direct gecorrigeerd, maar uit oogpunt van efficiëntie is het niet de bedoeling dat er wordt gediscussieerd over bevindingen en (mogelijke) oplossingen.

Tenslotte wordt bepaald of het product wordt geaccepteerd zoals het is (eventueel na enkele kleine wijzigingen), of dat

het product na wijzigingen en controle door één van de inspecteurs wordt geaccepteerd, of dat na wijzigingen een herinspectie op het product moet plaatsvinden.

Op basis van de tijdens de bijeenkomst gemaakte bevindingenlijst en afspraken past de auteur het product aan.

Exitcriteria

Het inspectieproces wordt als beëindigd beschouwd als:

- De wijzigingen (herstelwerkzaamheden) zijn afgerond (controle door moderator).
- Het product een nieuw versienummer heeft gekregen.
- Alle wijzigingen zijn gedocumenteerd in de nieuwe versie van het geïnspecteerde product (change history).
- Eventuele wijzigingsvoorstellen op andere producten, als resultaat van het inspectieproces, zijn ingediend.
- Het inspectieformulier volledig is ingevuld en overhandigd aan de verantwoordelijke kwaliteitszorgmedewerker ten behoeve van onder andere het opbouwen van statistieken.

15.4 Reviews

Er bestaan diverse reviewsoorten, zoals: technische review (o.a. kiezen van oplossingrichting/alternatief), management review (o.a. bepalen projectstatus), collegiale review (review door collega) en expert review (review door deskundigen). In deze paragraaf wordt een algemene invulling van het reviewproces gegeven.

Inleiding

Bij het uitvoeren van een review wordt een formele werkwijze gevolgd, waarbij een (60% tot 80% gereed) product aan een aantal reviewers wordt aangeboden, met de vraag of zij het willen beoordelen vanuit een bepaald aandachtsgebied (afhankelijk van de reviewsoort). De auteur verzamelt het commentaar en zal het product op grond daarvan bijwerken.

Een review is vooral gericht op het zoeken naar oplossingsrichtingen op basis van kennis en vaardigheden van de reviewers én op het vinden en corrigeren van fouten. Een review van een product vindt vaak vroeger in de levenscyclus van het product plaats dan een inspectie van het product.

Verantwoordelijkheden

Het minimale aantal deelnemers bij het uitvoeren van een review is drie. Bij een collegiale review kan dit soms minder zijn. Bijvoorbeeld bij een review van de code, welke doorgaans door één reviewer wordt uitgevoerd. Mogelijke rollen zijn:

- Moderator
  De moderator bereidt het reviewproces voor en leidt dit proces. Dit houdt in het plannen van het reviewproces, het uitnodigen van de reviewers en het eventueel toekennen van specifieke taken aan de reviewers.
- Auteur
  De auteur vraagt een review aan en levert het product ter review aan.
- Notulist
  De notulist legt tijdens het reviewproces alle bevindingen, acties, beslissingen en aanbevelingen van de reviewers vast.
- Reviewer
  De reviewer probeert voorafgaand aan de reviewbijeenkomst zoveel mogelijk bevindingen te vinden en te documenteren.

Alle deelnemers kunnen één of meer van bovenstaande rollen invullen, waarbij alle deelnemers (met uitzondering van de auteur) in ieder geval de rol van reviewer hebben.

Entrycriteria

Het reviewproces kan worden gestart als:

- Het doel van de review en de reviewprocedure duidelijk zijn.

- Het te reviewen product beschikbaar is, waarbij een uitgebreide ‘entry check’ niet nodig is, omdat het product pas voor 60%-80% gereed is.
- De bij de review te gebruiken reviewformulieren voor het vastleggen van bevindingen aanwezig zijn.
- (eventueel) Een lijst met al bekende aanwezige bevindingen beschikbaar is.

Procedures

Het reviewproces kan worden verdeeld in de fasen planning, voorbereiding en uitvoering:

Planning

Wanneer het te reviewen product aan de entrycriteria voldoet, wordt door de moderator een reviewbijeenkomst georganiseerd. Dit houdt onder andere in: datum en plaats voor de bijeenkomst bepalen, reviewers uitnodigen, te reviewen producten onder de deelnemers verspreiden en afspraken maken over de termijn waarop de auteur over de bevindingen moet kunnen beschikken.

Voorbereiding

Voor het houden van een zo effectief en efficiënt mogelijke reviewbijeenkomst, is een goede voorbereiding noodzakelijk. Tijdens de voorbereiding zoeken de reviewers naar bevindingen in de te reviewen producten en leggen deze vast in het reviewformulier. De moderator verzamelt deze formulieren, classificeert de bevindingen bij voorkeur en stelt het resultaat hiervan tijdig aan de auteur beschikbaar, zodat de auteur zich kan voorbereiden.

Uitvoering

Bij aanvang van de bijeenkomst wordt onder leiding van de moderator de agenda opgesteld of aangepast. De belangrijkste bevindingen worden bovenaan de agenda gezet. Doel hiervan is om genoeg tijd in de agenda te reserveren voor het kunnen bespreken van deze bevindingen. Aangezien het product nog niet gereed was, wordt er weinig tot geen aandacht besteed aan minder belangrijke bevindingen (dit in tegenstelling tot de uitvoering van het inspectieproces). De bevindingen worden door de notulist vastgelegd in een bevindingenlijst. Vaak wordt ook een actielijst opgesteld.

Tenslotte geeft de moderator eventueel een aanbeveling voor een aanvullende review. Dit wordt onder andere bepaald door de ernst en hoeveelheid van de bevindingen. Op basis van de tijdens de bijeenkomst gemaakte bevindingen-/actielijst past de auteur het product aan.

Exitcriteria

Het reviewproces wordt als beëindigd beschouwd als:

- Alle acties van de actielijst zijn ‘gesloten’ en alle wijzigingen naar aanleiding van belangrijke bevindingen zijn in het product verwerkt (controle door moderator).
- Het product goed is bevonden voor gebruik in een volgende fase/activiteit.

15.5 Walkthroughs

Het houden van een walkthrough is een werkwijze waarbij de auteur in een bijeenkomst uitlegt wat de inhoud van een product is. Hierbij zijn verschillende doelen mogelijk:

- Alle deelnemers naar hetzelfde uitgangsniveau brengen, bijvoorbeeld als voorbereiding op een review- of inspectieproces.
- Overdracht van informatie, bijvoorbeeld aan ontwikkelaars en testers om hen te helpen bij respectievelijk hun programmerings- en testontwerpwerkzaamheden.
- Aan de deelnemers vragen om aanvullende informatie.
- De deelnemers laten kiezen uit door de auteur aangedragen alternatieven.

Inleiding

Een walkthrough kan voor elk van de in paragraaf 15.2 genoemde documenten, die voor 50% tot 100% gereed zijn, worden gehouden.

Verantwoordelijkheden

Het aantal deelnemers bij een walkthrough is niet gelimiteerd in het geval de auteur zijn product bijvoorbeeld door middel van een presentatie aan bepaalde groepen wil toelichten. Bij een interactieve walkthrough is een groepomvang van twee tot zeven personen aan te bevelen. Mogelijke rollen zijn dan:

- Moderator
  De moderator bereidt de walkthrough voor. Dit houdt in het plannen van de walkthrough, het uitnodigen van de deelnemers, het verspreiden van zowel het product als een document met het doel van de walkthrough.
- Auteur
  De auteur vraagt om een walkthrough en licht het product tijdens de walkthrough toe.
- Notulist
  De notulist legt tijdens de walkthrough alle genomen beslissingen en geidentificeerde acties vast. Hiernaast worden bijvoorbeeld ook bevindingen (o.a. tegenstrijdigheden, vragen en omissies) en aanbevelingen van de deelnemers genotuleerd.
- Deelnemer
  De rol van deelnemer is afhankelijk van het doel van de walkthrough. Deze kan variëren van toehoorder tot het actief aandragen van bepaalde oplossingen.

Alle deelnemers kunnen één of meer van bovenstaande rollen invullen. De auteur kan de rol van moderator op zich nemen. Zowel de moderator als de auteur kan de rol van notulist invullen.

Entrycriteria

De walkthrough kan worden gestart als:

- Het doel van de walkthroughs duidelijk is.
- Het onderwerp (product) van de walkthrough beschikbaar is.

Procedures

Het walkthrough proces kan worden verdeeld in de fasen planning, voorbereiding en uitvoering:

Planning

Wanneer aan de entrycriteria is voldaan, worden door de moderator de walkthrough gepland, deelnemers uitgenodigd, het product verspreid en het doel van de walkthrough aan de deelnemers duidelijk gemaakt.

Voorbereiding

Afhankelijk van het doel van de walkthrough leveren deelnemers meestal geen (bijvoorbeeld als het doel kennisoverdracht is) of soms wel bevindingen aan. In het laatste geval worden de door de moderator verzamelde bevindingen aan de auteur ter beschikking gesteld. De auteur bepaalt hoe het product wordt toegelicht, bijvoorbeeld gerelateerd aan eventuele bevindingen, sequentieel, ‘bottom up’ of ‘top down’.

Uitvoering

Bij aanvang van de bijeenkomst wordt door de moderator het doel van de walkthrough en de te volgen procedure toegelicht. Vervolgens licht de auteur het product in detail toe en mogen de deelnemers tijdens of na afloop van de toelichting vragen stellen, op- en aanmerkingen maken, enzovoort.

Genomen beslissingen, geïdentificeerde acties, eventuele bevindingen, enzovoort worden door de notulist genotuleerd. Na afloop van de walkthrough neemt de moderator met alle aanwezigen de genotuleerde beslissingen, acties en andere belangrijke informatie met de deelnemers ter controle door.

Exitcriteria

De walkthrough wordt als beëindigd beschouwd als:

- Het product tijdens de walkthrough is toegelicht.
- Beslissingen, acties en aanbevelingen zijn genotuleerd.
- Het doel van de walkthrough is bereikt.

15.6 Keuzematrix toetstechnieken

Net als bij testen wordt bij iedere organisatie of bij ieder project het toetsen op een eigen manier ingevuld. Dit betekent dat er niet één uniforme beschrijving van de toetstechnieken is te geven en in welke situatie een bepaalde techniek het best is te gebruiken. Maar wellicht kan onderstaande tabel (ook te vinden op www.tmap.net) toch enigszins hulp bieden bij het kiezen van een techniek:

Aspect	Inspectie	Review	Walkthrough
Toepassingsgebied	Naast het toetsen of de oplossing goed is verwerkt, ook gericht op het verkrijgen van consensus over de kwaliteit van een product.	Vooral gericht op het zoeken naar oplossingsrichtingen en het vinden en corrigeren van fouten. Reviewsoorten zijn o.a.: technisch, management-, collegiale- en expert review.	Gericht op het kiezen uit alternatieve oplossingen, het invullen van ontbrekende informatie, of kennisoverdracht.
Te toetsen producten	Bijvoorbeeld: functioneel-/technisch ontwerp, requirementsdocument, managementplan, ontwikkelplan, testplan, onderhoudsdocumentatie, gebruikers-/installatie handleiding, software, release note, testontwerp, testscript, prototype, en screen print.		
Groepsomvang	Drie tot zes deelnemers.	Minimaal drie deelnemers	Van twee tot zeven deelnemers bij interactieve vorm tot een onbeperkt aantal bij presentatievorm.
Voorbereiding	Strakke sturing op welke aspecten door de inspecteurs moeten worden beoordelen. Bevindingen (op basis van checklists, standaarden, enz.) van inspecteurs vooraf aan de bijeenkomst bij auteur aanleveren.	Reviewers bepalen grotendeels zelf welke aspecten ze willen beoordelen. Bevindingen van reviewers vooraf aan bijeenkomst bij auteur aanleveren.	Van alleen kennisnemen van het tussenproduct tot het aanleveren van bevindingen.
Productstatus en -omvang	Product is 100% gereed, nog niet definitief en beperkt van omvang (1020 pagina's).	Product is 60%-80% gereed en variabel van omvang.	Product is 50%-100% gereed en variabel van omvang.
Voordelen	Kwalitatief hoog, incidentele- en structurele kwaliteitsverbetering.	Beperkt arbeidsintensief, vroege betrokkenheid reviewers.	Groot leereffect, beperkt arbeidsintensief.
Nadelen	Arbeidsintensief (duur), relatief lange doorlooptijd.	Subjectief, mogelijke verstoring collegiale verhoudingen.	Kans op ad hoc discussies, omdat deelnemers vaak onvoorbereid zijn.

Tabel 15.1: Keuzematrix toetstechnieken

16 Testrollen

16.1 Inleiding

Testwerkzaamheden in een project of in de lijn worden benoembaar gemaakt door taken te definiëren. Voor het uitvoeren van die taken zijn kennis en vaardigheden nodig. Een groep taken met de daarbij vereiste kennis en vaardigheden wordt een rol genoemd.

Definitie

Een rol is de beschrijving van één of meer taken met de hiervoor vereiste kennis en vaardigheden.

Er zijn rollen die één op één overeenkomen met de functies die in paragraaf 8.6.5 “Mogelijke functies van een testprofessional” beschreven zijn. Daarnaast zijn er rollen die niet voorkomen als functie.

Verschil functie en rol

Het belangrijke verschil tussen een functie en een rol is dat de professional een functie *heeft* en een rol *vervult*.

Een rol is gericht op het invullen van taken voor het testproject of de permanente testorganisatie in een specifieke situatie. Rollen worden vervuld door een testprofessional.

Een functie van een testprofessional beschrijft formeel welke taken deze moet kunnen uitvoeren en welke kennis en vaardigheden deze moet hebben. Een functie zegt iets over de geschiktheid voor een rol.

Voorbeeld 1

Een testproject wil de rol tester invullen. Daartoe wordt een persoon gezocht die de functie van tester heeft.

Voorbeeld 2

Een testproject zoekt invulling van de rol testteamleider met een persoon die ook moet gaan testen (meewerkend voorman). Daartoe wordt een iemand gezocht die de functie van tester heeft, beschikt over praktijkervaring als tester, beschikt over de capaciteit leiding geven en de wens heeft testteamleider te willen worden.

In dit hoofdstuk worden veelvoorkomende rollen in een testproject of in een permanente testorganisatie benoemd. Rollen die overeenkomen met testfuncties zijn benoemd in paragraaf 16.2. In paragraaf 16.3 worden de rollen beschreven die geen expliciete functiebeschrijving kennen.

16.2 Rollen die als functie zijn beschreven

Onderstaande rollen zijn reeds als functie beschreven in paragraaf 8.6.5 “Functies”.

- Tester
- Testtoolprogrammeur
- Methodisch testspecialist
- Testcoördinator
- Testtoolspecialist
- Testconsultant
- Testmanager
- Testtoolconsultant.

De rol van tester kan bij GAT en PAT behalve door een professional met de functie tester ook ingevuld worden door een gebruiker respectievelijk een beheerder.

16.3 Rollen niet als functie beschreven

In deze paragraaf zijn rollen beschreven die niet als functie zijn beschreven. Naast de hier beschreven rollen kunnen in de praktijk andere rollen voorkomen. Tevens kunnen rollen worden gecombineerd.

- Applicatie integrator
- Bevindingenbeheerder
- Intermediair voor bevindingen
- Materiedeskundige
- Systeemdeskundige
- Testinfrastructuurcoördinator
- Testprojectadministrateur
- Testteamleider
- Testwarebeheerder.

Per rol wordt toegelicht wat de taken en de vereiste kennis en vaardigheden zijn.

Applicatie integrator

De applicatie integrator is verantwoordelijk voor de integratie van de afzonderlijke systeemdelen (programma's, objecten, modules, componenten, enzovoort) tot een correct werkend systeem en kan fungeren als kwaliteitscontroleur om de gemaakte afspraken te controleren. De applicatie integrator ondersteunt de unittest en voert de unitintegratietest uit.

De applicatie integrator levert het systeem op aan de volgende fase in het project.

Taken

- Integreren van de losse programma's, objecten, modules, componenten, enzovoort, tot gebruikersfuncties of (deel)systemen;
- Opstellen, verkrijgen van goedkeuring en onderhouden van het unitintegratietestplan;
- Opstellen van entry-criteria waar de afzonderlijke programma's, objecten, modules, componenten aan moeten voldoen om opgenomen te worden in het integratieproces;
- Opstellen van exit-criteria waar een geïntegreerd deel van het systeem aan moet voldoen om vrijgegeven te kunnen worden aan de volgende fase;
- Faciliteren en ondersteunen van de programmeurs bij het uitvoeren van unittests;
- Uitvoeren van het unitintegratietestplan binnen planning en budget;
- (Doen) Uitvoeren van configuratie- en versiebeheer;
- (Doen) Uitvoeren van intern bevindingenbeheer;
- Fungeert als contact met de organisatie van de opdrachtgever met betrekking tot latere testsoorten;
- Rapporteren over voortgang van het integratieproces en de kwaliteit van het testobject;
- Opstellen vrijgaveadvies van de testsoort unitintegratietest;
- Evalueren integratieproces;
- Opleveren systeem aan de volgende fase.

Vereiste kennis en vaardigheden

Testspecifiek

- Ervaring in het toepassen van de TMap-fasering en testontwerptechnieken;
- Ervaring als teamleider en in ondersteuningsfuncties;

Automatisering

- Kennis van de systeemontwikkelmethode;
- Kennis van de architectuur en tools voor systeemontwikkeling;
- Kennis van de gebruikte hardware, software (inclusief programmataal) en datacommunicatiemiddelen; Kennis van integratietechnieken en integratietools

Algemeen

- Globale kennis van de materie en de lijnorganisatie;
- Goede contactuele eigenschappen en een motiverende uitstraling;
- Dwingt respect af bij ontwikkelaars;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid;
- Kritische instelling en kan daardoor argumenten op hun juiste waarde schatten;
- Tactvol, paniekbestendig en in staat kritiek te verdragen;
- Beschikt over het vermogen om te ondersteunen bij de foutanalyse.

Bevindingenbeheerder

De bevindingenbeheerder draagt zorg voor het inrichten en het optimale gebruik van de bevindingenadministratie en de processen daarom heen.

Taken

- Inrichten van de bevindingenadministratie en de bijbehorende processen;
- Adviseren over de selectie van een bevindingenbeheertool;
- Monitoren, evalueren en verbeteren van de gebruikswijze van de bevindingenbeheertool en de processen;
- Beheren van bevindingen op aangeven van het bevindingenoverleg en/of beslisforum;
- Beheren van de toegang tot en autorisaties binnen de bevindingenbeheertool.

Vereiste kennis en vaardigheden

Testspecifiek

- Kennis van de TMap-fasering;
- Uitgebreide kennis van bevindingenprocedures en de relatie met het testproces;

Automatisering

- Algemene kennis van automatisering, systeemontwikkeling en de projectaanpak;
- In staat nieuwe en aangepaste rapportages uit de bevindingenbeheertool te laten komen;

Algemeen

In staat procedures en projectdoelen te vertalen naar adequate processen.

Intermediair

De intermediair onderhoudt op uitvoerend niveau contact met testteam, ontwikkelaars en materiedeskundigen/gebruikers over de bevindingen. Enerzijds licht de intermediair de bevindingen toe en zorgt indien gewenst voor aanvullende gegevens, anderzijds geeft deze de oplossingen van de bevindingen door aan het testteam.

Deze rol behoeft nadrukkelijk expliciete aandacht indien het testteam zich op een andere locatie bevindt of indien er sprake is van zeer formele verhouding met de andere partijen, bijvoorbeeld bij outsourcing.

Taken

- Toelichten bevindingen aan oplossers en deelnemers van het bevindingenoverleg;

- Dubbele bevindingen filteren;
- Bevindingen van aanvullende gegevens (laten) voorzien;
- Laten toetsen of bevindingen in andere omgevingen herhaalbaar zijn;
- Controleren oplossing op volledigheid en op het inpassen in het projectbeleid;
- Toelichten oplossing van bevindingen;
- Bewaken van de status van bevindingen;
- Rapporteren over de voortgang van oplossingen mede in het licht van eventuele afspraken over de oplossingsnelheid;
- Adviseren van overlegorganen over bevindingen.

Vereiste kennis en vaardigheden

Testspecifiek

- Globale kennis van de TMap-fasering;

Automatisering

- Algemene automatiseringskennis en -ervaring;
- Vaardigheid in het interpreteren van de testbasis in iedere vorm;

Algemeen

- Goede kennis van de afspraken in het project voor het prioriteren van bevindingen;
- Uitstekende contactuele eigenschappen;
- Uitstekende schriftelijke en mondelinge uitdrukkingsvaardigheid;
- Vaardig in conflicthantering en onderhandelingstechnieken;
- In staat kritiek te verdragen.

Materiedeskundige

De materiedeskundige zorgt voor de inbreng van materiekennis en inzicht hoe deze materie gespecificeerd is in requirements, functionele ontwerpen, bedrijfsprocessen en gebruikershandleidingen.

Taken

- Verlenen van ondersteuning en advies over functionaliteit en bedrijfsprocessen bij:
  - Analyseren productrisico's;
  - Bepalen teststrategie;
  - Introductie nieuwe testmedewerkers;
  - Doen van aannames bij bevindingen op het gebied van consistentie;
  - Analyseren van bevindingen;
  - Prioriteren van bevindingen;
  - Opstellen en/of reviewen van testgevallen.

Vereiste kennis en vaardigheden

Testspecifiek

- Globale kennis van de TMap-begrippen;

Automatisering

- Algemene automatiseringskennis en -ervaring;
- Kennis van de functionele architectuur principes van het testobject;

Algemeen

- Specialistische kennis van de materie, zowel de bedrijfsprocessen als de functionaliteit;
- Kennis van de organisatie waar het testobject in gebruik gaat functioneren;
- Goede contactuele eigenschappen en een motiverende uitstraling;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid.

Systeemdeskundige

De systeemdeskundige heeft de verantwoordelijkheid de testers de functionaliteit en globale technische architectuur van het te testen systeem toe te lichten. Hij/zij zal ondersteuning verlenen bij systeemtest, functionele acceptatietest, gebruikersacceptatietest en systeemintegratietest.

Taken

- Toelichten functionaliteit van het systeem mede in relatie tot de bedrijfsprocessen;
- Toelichten van de technische architectuur van het systeem.

Vereiste kennis en vaardigheden

Testspecifiek

- Globale kennis van de TMap-fasering;

Automatisering

- Gedetailleerde kennis van de functionaliteit;
- Gedetailleerde kennis van de technische architectuur;
- Vaardigheid in het interpreteren van de functionele specificaties;

Algemeen

- Globale kennis van de materie;
- Goede contactuele eigenschappen.

Testinfrastructuurcoördinator

De testinfrastructuurcoördinator is binnen het testteam verantwoordelijk voor de contacten met de partijen die belast zijn met het inrichten en behouden van een hoog niveau van beschikbaarheid van testinfrastructuur: testomgeving, testtools en werkplekken voor de tester.

Deze rol wordt bij voorkeur door één persoon ingevuld, en is vaak parttime.

Indien de geschikte persoon niet te vinden is, kunnen de taken in de rol worden verdeeld over meerdere personen.

Taken

- Zorg dragen voor een tijdige beschikbaarstelling van de testomgeving;
- Zorg dragen voor een tijdige beschikbaarstelling van testtools, waarbij
- planning- en beheertools eerder nodig zijn dan testuitvoeringstools;
- Zorg dragen voor een tijdige beschikbaarstelling van werkplekken;
- Afstemmen met andere partijen over testinfrastructuur;
- Advisering over:
  - Oplossen bevindingen/incidenten op gebied van testinfrastructuur;
  - Impact wijzigingvoorstel testinfrastructuur;
  - Specificaties testinfrastructuur;

– Kosten van het gebruik van testinfrastructuur.

Vereiste kennis en vaardigheden

Testspecifiek

- Kennis van de TMap-fasering;
- Kennis van testtools;

Automatisering

- Algemene kennis van hardware, omgevingssoftware, koppelingen en omgevingdata;
- Specifieke kennis van hardware, omgevingssoftware, koppelingen en omgevingdata die voor het te testen systeem nodig zijn;
- Kennis van de exploitatie(=gebruik) en beheer van het systeem;

Algemeen

- Kennis van de processen in de organisatie die zorgen voor beschikbaarheid van testomgeving, testtools en werkplekken, zowel voor structurele als incidentoplossende zaken;
- Goede contactuele eigenschappen;
- Goede schriftelijke en mondelinge uitdrukkingsvaardigheid;
- Actief in het (laten) oplossen van incidenten.

Testprojectadministrateur

De testprojectadministrateur verzorgt de administratie van de testprojectgegevens en de beschikbaarstelling van deze gegevens. Gegevens kunnen zowel voortgangsgegevens als projectdocumentatie omvatten.

Taken

- Verzamelen, controleren en registreren van voortganggegevens;
- Beschikbaar stellen van voortganggegevens;
- Verzamelen en administreren van projectdocumentatie.

Vereiste kennis en vaardigheden

Testspecifiek

- Globale kennis van de TMap-fasering;

Automatisering

- Kennis van spreadsheet- en tekstverwerkingsprogramma's;

Algemeen

- Uitstekende administratieve vaardigheid;
- Vaardigheid in het gebruik van projectmanagementtools;
- Nauwgezet.

Testteamleider

De rol testteamleider omvat de verantwoordelijkheid voor het dagelijks werk van een aantal testers.

Deze rol wordt geïntroduceerd als het testteam gesplitst wordt in meer teams.

Redenen voor splitsing kunnen zijn: het testteam wordt te groot voor één testmanager of het testteam opereert op meer locaties. Ieder team wordt dan geleid door een testteamleider.

In een klein testteam wordt deze rol gecombineerd met de rol tester in de combinatierol meewerkend voorman.

Taken

- Opstellen detailplanning testteam;
- Bewaken van en rapporteren over de voortgang van het testproces;
- Rapporteren over de kwaliteit van het testobject;
- Doen oplossen van incidenten rondom testinfrastructuur;
- Rapporteren knelpunten met waar mogelijk oplossingen;
- Dagelijkse leiding geven aan testteam;
- Coaching van testers in het testteam.

Vereiste kennis en ervaring

Testspecifiek

- Uitgebreide kennis van de TMap-fasering en de primaire testactiviteiten;
- Specialistische vaardigheid in de detail intake en de testontwerptechnieken;
- Kennis van het gebruik van testtools;

Automatisering

- Algemene automatiseringskennis en -ervaring;
- Kennis van systeemontwikkelingtechnieken;
- Vaardigheid in het interpreteren van testbasis;

Algemeen

- Vaardigheid in het leidinggeven;
- Kennis van projectmatig werken;
- In staat zijn te coachen.

Testwarebeheerder

De testwarebeheerder is verantwoordelijk voor het beheer van de testware en controleert de naleving van de testvoorschriften op het gebied van testware.

Taken

- Opzetten testwarebeheer, procedures en voorschriften;
- Beheer van de testware;
- Fysiek configuratiemanagement van testware, ook in relatie tot het configuratiebeheer van testbasis, testobject en testinfrastructuur;
- Controleren naleving voorschriften voor testware.

Vereiste kennis en vaardigheden

Testspecifiek

- Kennis van de TMap-fasering;
- Globale kennis van de soorten testware;
- Kennis van de testvoorschriften voor testware;

Automatisering

- Algemene kennis van het systeemontwikkelproces en systeemontwikkelproducten;
- Kennis van beheertools voor testwarebeheer en configuratiemanagement;

Algemeen

- Pro-actieve houding;
- Nauwgezet en strikt volgens de procedures te werk gaan;
- Creatief in het bedenken van oplossingen binnen de gestelde regels;
- Goede contactuele eigenschappen.

Woordenlijst

Acceptatietest
De door de toekomstige gebruiker(s) en beheerder(s) in een zoveel mogelijk als-ware-het-productie omgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de functionele en kwalitatieve eisen voldoet.

Basistechniek
Basistechniek is de wijze van afleiden van de testsituaties uit de testbasis die tot de gewenste dekkingsvorm leidt.

BDTM-aspecten
Resultaat, Risico, Kosten, Tijd.

Bedrijfszekerheid
De mate waarin het informatiesysteem vrij blijft van storingen.

Beheerbaarheid
Het gemak waarmee het informatiesysteem in operationele staat kan worden gebracht en gehouden.

Beveiliging
De zekerheid dat raadpleging of mutatie van de gegevens uitsluitend mogelijk is door die personen die daartoe bevoegd zijn.

Bevinding
Een bevinding is een geconstateerd verschil tussen de verwachting of voorspelling en de feitelijke uitkomst.

Bruikbaarheid
De mate waarin het informatiesysteem is toegesneden op de organisatie en het profiel van de eindgebruikers voor wie het bedoeld is en bijdraagt aan het bereiken van de bedrijfsdoelstellingen.

Centrale uitgangssituatie
Een uitgangssituatie die voor meer tests of testers de uitgangssituatie bevat.

Connectiviteit
Het gemak waarmee een koppeling met een ander informatiesysteem of binnen het informatiesysteem tot stand kan worden gebracht.

Continuïteit
De zekerheid dat de gegevensverwerking ongestoorde voortgang zal kunnen vinden, dat wil zeggen ook na ernstige storingen binnen redelijke termijn kan worden hervat.

Controleerbaarheid
Het gemak waarmee de juistheid en volledigheid van de informatie (in het verloop van de tijd) gecontroleerd kan worden.

Degradatiemogelijkheid
Het gemak waarmee de kern van de informatievoorziening kan worden voortgezet nadat een deel is uitgevallen.

Dekking
Dekking is de verhouding tussen datgene wat getest kan worden en datgene wat met de testset getest wordt.

Dekkingsgraad
Dekkingsgraad is het percentage van de door de dekkingsvorm bepaalde testsituaties dat door de test gedekt is.

Dekkingsvorm
Dekkingsvorm is de vorm waarin het afdekken van de te testen situaties die afleidbaar zijn uit de testbasis uitgedrukt wordt.

Detail intake testbasis
Het in detail beoordelen van de testbasis op de testbaarheid.

Driver
Een simulatieprogramma dat een programma vervangt dat de sturing en aanroep van het testobject verzorgt.

Dynamisch testen
Testen, op basis van gerichte testgevallen, door executie van het testobject c.q. het draaien van programma's.

End-to-end test
Zie ketentest.

Equivalentieklasse
Bij het toepassen van equivalentieklassen wordt het volledige waardebereik van een parameter opgedeeld (gepartitioneerd) in klassen waarbij het systeemgedrag gelijksoortig (equivalent) is.

Exploratory Testing
Het simultaan leren, ontwerpen en uitvoeren van tests, met andere woorden elke vorm van testen waarbij de tester zijn tests ontwerpt tijdens de testuitvoering en de informatie die wordt verkregen tijdens het testen wordt gebruikt om nieuwe en betere testgevallen te ontwerpen.

Flexibiliteit
De mate waarin de gebruiker zelf uitbreidingen of variaties op het informatiesysteem kan aanbrengen zonder dat de programmatuur wordt aangepast.

Functiepunt
Meeteenheid voor de functionaliteit c.q. omvang van de applicatie-software.

Functiepuntanalyse
Functiepuntanalyse (FPA) is een methode met de mogelijkheid om een technologie-onafhankelijke meting te doen van de omvang van de door een geautomatiseerd systeem geboden functionaliteit en deze meting te gebruiken als basis voor productiviteitsmeting, het schatten van de benodigde middelen en projectbeheersing.

Functionaliteit
De zekerheid dat de verwerking van de gegevens juist en volledig geschiedt, conform de beschrijving in de functionele specificaties.

Functionele acceptatietest
De functionele acceptatietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de functionele eisen voldoet.

Fysiek testgeval
Een fysiek testgeval is de concrete uitwerking van een logisch testgeval, waarbij keuzes gemaakt zijn voor de waarden van alle benodigde invoer en instellingen van de omgevingsfactoren.

Gebruikersacceptatietest
De gebruikersacceptatietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de wensen/eisen van de gebruiker voldoet.

Gebruikersvriendelijkheid
Het gemak waarmee de eindgebruiker kan leren omgaan met het informatiesysteem en het bedieningsgemak van het informatiesysteem voor ingeleerde gebruikers.

Gecombineerde test
Testaanpak waarbij de systeemtest en de functionele acceptatietest tot één testsoort gecombineerd worden.

Generieke Test Afspraken
De projectoverstijgende algemene afspraken over bijvoorbeeld het testproces, de standaard strategie, de manier van begroten, de procedures, de organisatie, de communicatie, de documentatie, etc.

Geschiktheid infrastructuur
De geschiktheid van de apparatuur, het netwerk, de systeemsoftware en het DBMS voor de betreffende toepassing en de mate waarin deze infrastructuurelementen op elkaar aansluiten.

Grenswaardenanalyse
Testprincipe gebaseerd op het feit dat een test rondom de grenswaarde een grotere kans voor het vinden van een fout heeft.

Herbruikbaarheid
De mate waarin delen van het informatiesysteem, of van het ontwerp, opnieuw gebruikt kunnen worden voor de ontwikkeling van andere toepassingen.

Herstelbaarheid
Het gemak en de snelheid waarmee de informatievoorziening na een storing hersteld kan worden.

Initiële situatie
De initiële situatie omvat alles wat nodig is om het systeem klaar te zetten om de bedoelde input te ontvangen. Hieronder vallen niet alleen de gegevens die voor de 'processing' nodig zijn, maar ook de toestand waarin het systeem en zijn omgeving zich in moeten bevinden. Denk bijvoorbeeld aan het instellen van een bepaalde systeemdatum, of het draaien van bepaalde week- en maand-batches die het systeem in een bepaalde toestand brengen.

Inpasbaarheid
De mate waarin de handmatige procedures aansluiten op het geautomatiseerde informatiesysteem en de werkbaarheid van deze handmatige procedures voor de organisatie.

Juistheid
De mate waarin het systeem de aangeboden invoer en mutaties correct volgens de specificatie verwerkt tot consistente gegevensverzamelingen.

Ketentest
Een testtype dat end-to-end functionaliteit van één of meer systemen test met testgevallen die buiten het systeem beginnen en buiten het systeem eindigen.

Known error
Een bevinding die onderkend is als daadwerkelijke fout, maar (nog) niet opgelost worden.

Kwaliteit
Kwaliteit is het geheel van eigenschappen en kenmerken van een product of dienst dat van belang is voor het voldoen aan vastgestelde of vanzelfsprekende behoeften.

Kwaliteitsattribuut
Een kwaliteitsattribuut beschrijft een kenmerk van een informatiesysteem.

Kwaliteitsborging
Het geheel van alle geplande en systematische acties nodig om in voldoende mate het vertrouwen te geven dat een product of dienst voldoet aan de gestelde kwaliteitseisen.

Logisch testgeval
Een logisch testgeval beschrijft in logische termen de omstandigheden waarin het systeemgedrag onderzocht wordt, door aan te geven welke testsituaties door het testgeval gedekt worden.

Mastertestplan
Een testplan waarin de diverse testsoorten op elkaar afgestemd zijn.

Onderhoudbaarheid
Het gemak waarmee het informatiesysteem kan worden aangepast aan nieuwe wensen van de gebruiker, de veranderende externe omgeving of om fouten te herstellen.

Orthogonale array
Een orthogonale array LN (sk, t) is een 2-dimensionale array van N rijen en k kolommen, bestaande uit elementen die s waarden kunnen aannemen, waarbij iedere combinatie van t kolommen alle combinaties van de s waarden in gelijke hoeveelheid bevat.

Pairwise testing
Test alle mogelijkheden van elke willekeurige combinatie van 2 factoren.

Performance
De snelheid waarmee het informatiesysteem interactieve en batch-transacties afhandelt.

Permanente testorganisatie
De permanente testorganisatie is een lijnorganisatie die testdiensten aanbiedt.

Portabiliteit
De diversiteit van het hardware- en software-platform waarin het informatiesysteem kan draaien, en het gemak waarmee het systeem kan worden overgebracht van de ene omgeving naar een andere.

Pretest
Het zodanig testen van de opgeleverde producten, dat bepaald wordt of het zinvol is een gestructureerde test met betrekking tot het testobject uit te voeren.

Productieacceptatietest
De productieacceptatietest is een door de toekomstige beheerder(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de van uit beheer gestelde eisen voldoet.

Productrisico
Een productrisico is de kans dat het product faalt in relatie tot de verwachte schade wanneer dit optreedt.
Productrisico = Faalkans * Schade, waarbij Faalkans = Foutkans * Frequentie van gebruik.

Productrisicoanalyse
De productrisicoanalyse is het analyseren van het te testen product met als doel dat testmanager en de verschillende andere belanghebbenden tot een gezamenlijk beeld komen van wat de meer of minder risicovolle kenmerken en onderdelen van het te testen product zijn, zodat de grondigheid van testen hieraan gerelateerd kan worden.

Regressie
Regressie is het verschijnsel dat de kwaliteit van een systeem als geheel terugloopt als gevolg van individuele aanpassingen.

Regressietest
Een regressietest is erop gericht om te controleren dat alle ongewijzigde onderdelen van een systeem nog correct functioneren na het doorvoeren van een wijziging.

Robuustheid
De mate waarin de informatievoorziening ook na een storing gewoon door kan gaan.

Simulator
Een simulator bootst de werking van de omgeving van het te testen (deel van het) systeem na.

Statisch testen
Testen door het controleren en onderzoeken van producten, zonder dat er sprake is van het uitvoeren van programma's.

Stub
Een simulatieprogramma dat een programma vervangt, inclusief de bijbehorende in- en uitvoerstromen, en wordt aangeroepen door het testobject.

Systeemintegratietest
Een systeemintegratietest is een door de toekomstige gebruiker(s) in een zoveel mogelijk als-ware-het-productieomgeving uitgevoerde test, die moet aantonen dat (sub)systeeminterface afspraken zijn nagekomen, correct zijn geinterpreteerd en correct zijn geïmplementeerd.

Systeemtest
Een systeemtest is een door de leverancier van de oplossing in een (goed beheersbare) laboratoriumomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem of delen daarvan aan de functionele- en niet-functionele specificaties en het technisch ontwerp voldoen.

Test pattern
Een test pattern is een algemene oplossing voor een specifiek terugkomend testprobleem.

Testactie
Een handeling in een van te voren gedefinieerde uitgangssituatie die goed of fout kan verlopen.

Testbaarheid
Het gemak en de snelheid waarmee de functionaliteit en het prestatieniveau van het systeem (na iedere aanpassing) getest kunnen worden.

Testbasis
De testbasis is de informatie die het gewenste systeemgedrag definieert.

Testbeleid
Het testbeleid beschrijft hoe een organisatie omgaat met de mensen, middelen en methoden rondom het testproces in de verschillende situaties.

Testdoel
Een testdoel is een voor de opdrachtgever relevant doel voor het testen, vaak geformuleerd in termen van door IT ondersteunde bedrijfsprocessen, gerealiseerde user requirements of use cases, kritische succesfactoren, wijzigingsvoorstellen of benoemde (af te dekken) risico's.

Testeenheid
Een verzameling processen, transacties en/of functies die gezamenlijk worden getest.

Testen
Testen is een proces dat inzicht geeft in- en adviseert over de kwaliteit en de daaraan gerelateerde risico's.

Testgeval
Met een testgeval wordt onderzocht of het systeem onder bepaalde omstandigheden het gewenste gedrag vertoont.

Testinfrastructuur
De testinfrastructuur bestaat uit de faciliteiten en middelen die nodig zijn om de test adequaat te kunnen uitvoeren. Er wordt onderscheid gemaakt tussen testomgevingen, testtools en werkplekken.

Testmaat-N
Testmaat-N is de zekerheid dat alle combinaties van N achtereenvolgende paden afgedekt zijn.

Testobject
Het te testen (deel van het) informatiesysteem.

Testomgeving
Een testomgeving is een samenstelling van onderdelen zoals hard- en software, koppelingen, omgevingsdata, beheertools en processen waarin een test wordt uitgevoerd.

Testontwerptechniek
Een testontwerptechniek is een gestandaardiseerde werkwijze om vanuit een bepaalde testbasis testgevallen af te leiden die een bepaalde dekking bereiken.

Testorganisatie
Een testorganisatie is het scheppen van doelmatige verhoudingen tussen testfuncties, testfaciliteiten en testactiviteiten teneinde tijdig een goed kwaliteitsadvies uit te brengen.

Testplan
In een testplan worden de globale opzet en de strategische keuzes met betrekking tot de uit te voeren test vastgelegd. Het testplan vormt het referentiekader gedurende de uitvoering van de test en dient tevens als instrument om met de opdrachtgever van de test te communiceren. Het testplan is een beschrijving van het testproject, inclusief een beschrijving van de activiteiten en de planning; dus niet een beschrijving van de tests zelf.

Testproces
Het geheel van activiteiten, procedures en hulpmiddelen om het testen uit te voeren.

Testpunt
Meeteenheid voor de omvang van een uit te voeren test.

Testraamwerk
Een testraamwerk (test harness) is een voor een ontwikkelomgeving geconfigureerde verzameling software en testgegegevens om één unit of serie van units dynamisch te testen waarbij het gedrag en de output wordt gecontroleerd.

Testscript
Opeenvolging van samenhangende acties en controles, gerelateerd aan fysieke testgevallen, waarvan de volgorde van uitvoering is aangegeven. Een beschrijving hoe er getest gaat worden.

Testset
Verzameling testgevallen die voor een bepaald testscript of testvorm wordt uitgevoerd.

Testsituatie
Een testsituatie is een geïsoleerde omstandigheid waaronder het testobject een specifiek gedrag vertoont en die getest moet worden.

Meeteenheid voor de omvang van een uit te voeren black-box test.

Testsoort
Een testsoort is een groep van testactiviteiten die gezamenlijk worden uitgevoerd en aangestuurd.

Testspecificatie
Een beschrijving van de wijze waarop de logische testgevallen zijn geselecteerd, alsmede een beschrijving van de logische testgevallen. Een beschrijving wat er getest gaat worden.

Teststraat
Een teststraat is de operationele organisatie voor het leveren van testdiensten met betrekking tot één of meer opdrachtgevers. In een teststraat zit een vast team testers, infrastructuur, testtools en gestandaardiseerde werkprocedures.

Teststrategie
De teststrategie is de verdeling van de testinspanning en dekkingsgraad over de te testen delen of aspecten van het testobject, met als oogmerk de belangrijkste fouten zo vroeg en goedkoop mogelijk te vinden. Deze ver deling is afhankelijk gemaakt van risico’s op het gebied van business, systeemontwikkeling en testen.

Testteam
Een samenwerkingsverband dat, onder leiding van een testmanager, testcoördinator of testteamleider testactiviteiten voor haar rekening neemt.

Testtechniek
Een testtechniek is een samenstel van acties om op universele wijze een testproduct te produceren.

Testtool
Een testtool is een geautomatiseerd hulpmiddel dat ondersteuning biedt aan één of meer testactiviteiten, zoals plannen, beheren, specificeren en uitvoeren.

Testvorm
Een groep testactiviteiten met het oogmerk het informatiesysteem op een aantal samenhangende kwaliteitsattributen te controleren.

Testware
Alle testdocumentatie, zoals testspecificaties, testscripts, een beschrijving van de testinfrastructuur, etc., die tijdens het testproces wordt geproduceerd. Als eis geldt dat deze testdocumentatie voor onderhoudsdoeleinden gebruikt moet kunnen worden en daarom overdraagbaar en onderhoudbaar moet zijn.

Tijdigheid
De mate waarin de informatie op tijd beschikbaar komt om de maatregelen te nemen waarvoor die informatie is bedoeld.

Toetsen
Toetsen is het beoordelen van de tussenproducten in het systeemontwikkelingsproces.

Toetssoort
Een toetssoort is een groep van toetsactiviteiten die gezamenlijk worden uitgevoerd en aangestuurd.

Toolbeleid
Het toolbeleid beschrijft hoe een organisatie omgaat met aanschaf, de invoering en het gebruik van testtools in de verschillende situaties.

Uitgangssituatie
Een uitgangssituatie voor een test bestaat uit een verzameling gegevens in en een begintoestand van het te testen (deel van het) systeem waar individuele testgevallen gebruik van kunnen maken. Er is sprake van een centrale uitgangssituatie als deze geldt voor meerdere tests of testers.

Uitwijkmogelijkheid
Het gemak waarmee (een deel van) de informatievoorziening op een andere locatie kan worden voortgezet.

Unitintegratietest
Een unitintegratietest is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een logische groep units aan de in de technische specificaties gestelde eisen voldoet.

Unittest
Een unittest is een door de ontwikkelaar in de ontwikkelomgeving uitgevoerde test, die moet aantonen dat een unit aan de in de technische specificaties gestelde eisen voldoet.

Volledigheid
De zekerheid dat alle invoer en mutaties verwerkt worden door het systeem.

Zuinigheid
De verhouding tussen het prestatieniveau van het systeem (uit te drukken in transactievolume en de totale snelheid) en de hoeveelheid resources die daarvoor gebruikt worden.

Referentielijst

Uit onderstaande publicaties is geput bij het schrijven van het boek.

- [Aalst, 1999]
  Aalst, L. van der, Koning, C. de, *Testen duur? Niet testen is duurder!*, Informatie oktober 1999, www.informatie.nl
- [Abran, 2003]
  Abran, A. (2003), *Implementation Guide for ISO/IEC 19761:2003 (Measurement Manual)*, version 2.2, Common Software Measurement International Consortium, www.cosmicon.com
- [Bach, 2000]
  Bach, Jonathan, (2000) Session-based test management, www.satisfice.com
- [Bach, 2002]
  Bach, James, (2002) Exploratory testing explained, www.satisfice.com
- [Bach, 2003]
  Bach, James. (1996-2002), Uit zijn workshop Rapid Software Testing
- [Basili, 1994]
  Basili, V.R., Galdiera, G. and Rombach, H.D. (1994), *The Goal-Question-Metric Approach*, John Wiley & Sons, Inc., New York
- [Basili, 1997]
  Basili, V., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S., Zelkowitz, M., *The Empirical Investigation of Perspective-Based Reading*, also appeared in: Empirical Software Engineering, 2, 1997
- [Beck, 2002]
  Beck, Kent, Test-Driven Development By Example (2002) Addison-Wesley Professional, ISBN 0321146530
- [Beizer, 1990] Beizer, B. (1990), *Software Testing Techniques*, International Thomson Computer Press.
- [Belbin, 2003]
  Belbin, R Meredith, Management Teams – Why They Succeed or Fail (second edition) (2003), Butterworth Heinemann, ISBN: 0 7506 5910 6
- [Bieberstein, 2005]
  Bieberstein, Norbert; Bose, Sanjay; Fiammante, Marc; Jones, Keith and Shah, Rawn, *Service-Oriented Architecture (SOA) Compass, Business Value, Planning and Enterprise Roadmap*, (2005) IBM Press, ISBN 0-13-187002-5
- [Black, 2002]
  Rex Black, keynote op Eurostar2002, Kopenhagen
- [Black, 2004]
  Black, Rex, *Critical Testing Processes* (2004) Addison Wesley, ISBN 0-201
- [Boehm, 1981]
  Boehm, B.W. (1981), *Software Engineering Economics*, Prentice-Hall Inc., Englewood Cliffs, ISBN 0-13-822122-7
- [Broekman, 2001]
  Broekman, Bart, Christiaan Hoos, Mark Paap, *Automatisering van de testuitvoering*, SDU, ISBN: 9 440 01030 5
- [Broekman, 2003]
  Broekman, B., Notenboom, E. (2003), *Testing Embedded Software*, AddisonWesley, London, ISBN 0-321-15986-1
- [Brooks, 1975/1995]
  Brooks, Frederic P,. The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition, (1975/1995), Addison-Wesley Professional, ISBN 0201835959
- [Clifton, 2004]
  Clifton, Marc, Advanced Unit Testing (2004), http://www.codeproject.com/gen/design/autp5.asp
- [Cockburn, 2000]
  Cockburn, Alistair and Williams, Laurie, The Costs and Benefits of Pair Programming, Humans and Technology Technical Report 2000.01, Jan 2000, http://alistair.cockburn.us
- [Collard, 1999]
  Collard, R. (1999), *Test Estimation*, STAREast 1999
- [Copeland, 2003]
  Copeland, L. (2003), *A Practitioner's Guide to Software Test Design*, Artech House Publishers, ISBN 1-58053-791-X
- [Deming, 1992]
  Deming, W. Edwards (1992), *Out of the crisis*, University of Cambridge, ISBN 0-521-30553-5
- [Fagan, 1986]
  Fagan, M.E. (1986), *Advances in Software Inspections*, in: IEEE Transactions on Software Engineering, July 1986
- [Hedayat, 1999]

Hedayat, A.S., Sloane, N.J.A. and John Stufken (1999) *Orthogonal arrays: Theory and Applications*, Springer Verlag, ISBN 0-387-98766-5
- [Hendriks, 1999]
  Hendriks, C.F.W. (1999), *Stempelend naar kwaliteit*, in: Computable nr 47, November 1999
- [IEEE, 1998]
  The Institute of Electrical and Electronics Engineers, Inc. (1998), *IEEE Std 1028-1997 Standard for Software Reviews*, 345 East 47th Street, New York, NY 10017-2394, USA, ISBN 1-55937-987-1
- [IFPUG, 1994]
  IFPUG (International Function Point User Group) (1994), *Function Point Counting Practices*, release 4.0, IFPUG, January 1994
- [ISO/IEC, 1991]
  ISO/IEC Guide 2 (1991), *General terms and definitions concerning standardization and related activities*, International Organization of Standardization
- [ISO 9126-1, 1999]
  ISO/IEC 9126 part 1 (1999), *Information Technology - Software Product Quality - Part 1: Quality Model*, International Organization of Standardization
- [ISO, 1994]
  ISO 8402 (1994), *Quality Management and quality assurance - vocabulary*, International Organization of Standardization
- [Jones, 1996]
  Jones, Capers, (1996), *Applied Software Measurement: Assuring Productivity & Quality*, McGraw Hill Text, ISBN 0070328269
- [Kaner, 1999]
  Kaner, C., Falk, J., Nguyen, H.Q. (1999), Testing computer software, 2nd edition, Wiley, ISBN 0-471-35846-0
- [Kaner, 2001]
  Kaner C., Bach J. (2001), *Exploratory testing in pairs*, StarEast 2001
- [Koomen, 2006]
  Koomen, T., Pol, M. (2006), *Test Process Improvement, Leidraad voor stapsgewijs beter testen*, Academic Service, Den Haag, ISBN 90-12-11360-1
- [Koomen, 2004]
  Koomen, T., Baarda, R., (2004), *TMap® Test Topics* (deel I), Tutein Nolthenius, 's-Hertogenbosch, 1ste druk, ISBN 90-72194-70-5
- [Laitenberger, 1995]
  Laitenberger, O., (1995), *Perspective based Reading: Technique, Validation and Research in Future*, Student project (Projektarbeit), Department of Computer Science, University of Kaiserslautern, 67653 Kaiserslautern, Germany
- [Lyndsay, 2002]
  Lyndsay, James(2002), Adventures in session-based testing, Proceedings EuroSTAR 2002
- [McCabe, 1976]
  McCabe, T.J.(1976*), A complexity metric*, IEEE Transactions on software engineering, vol. 2, IEEE Press
- [McCall, 1977]
  McCall, J.A., P.K. Richards en G.F. Walters (1977), *Factors in software quality*, RADC-TR-77-363 Rome Air Development Center, Griffis Air Force, Rome (New York, USA)
- [Musa, 1998]
  Musa, J.D. (1998), *Software Reliability Engineering*, McGraw-Hill
- [NESMA, 1996]
  NESMA (NEderlande Software Metrics Associatie) (1996), *Definities en telrichtlijnen voor de toepassing van functiepuntanalyse*, versie 2.0, NESMA, Amsterdam, april 1996
- [Nielsen, 1999]
  Jakob Nielsen, Designing Web Usability: The Practice of Simplicity, (1999) New Riders Publishing, Indianapolis, ISBN 1-56205-810-X
- Nielsen Jakob, www.useit.com, 2006
- [Pettichord, 2000]
  Pettichord, Bret, Testers and Developers Think Differently; Understanding and utilizing the diverse traits of key players on your team, Jan/Feb 2000, STQE Magazine
- [PRINCE2, 2002]
  Managing Successful Projects with PRINCE2 (2002), The Stationary Office London, ISBN 0-11-330891-4
- [Roden, 2005]
  Lloyd Roden, Choosing and Managing the Ideal Test Team, winter 2005 issue, www.methodsandtools.com
- [Rothman, 2006]

Johanna Rothman, “Hiring The Best Knowledge Workers, Techies & Nerds: The Secrets & Science Of Hiring Technical People, (2006) Dorset House Publishing Company, ISBN 0932633595
- [Schaefer, 1996] Schaefer, H., (1996), *Surviving under time and budget pressure*, in Proceedings EuroSTAR Conferencce 1996, Amsterdam
- [Splaine, 2002]
Splaine, Steven, Testing Web Security: Assessing the Security of Web Sites and Applications (2002) Wiley, ISBN 0471232815
- [The Standish Group, 2003]
The Standish Group, (2003), *What Are Your Requirements?*, The Standish Group International, West Yarmouth
- [Testkubus, 2005]
De TestKubus, Expertisegroep Testkubus, 1.0, april 2005, www.tmap.net
- [Tufte, 2001]
Edward Tufte, The visual display of quantitative information, 2nd edition, 2001, Graphics Press LLC, ISBN 0961392142
- [Vaaraniemi, 2003]
Vaaraniemi, Sami, The benefits of automated unit testing, November 2003, http://www.codeproject.com
- [Whittaker, 2000]
Whittaker, J., Jorgenson, A. (2000), *How to break software*, proceedings Eurostar 2000
- [Whittaker, 2003]
Whittaker, James A., Thompson, Herbert H., *How to Break Software Security* (2003) Addison Wesley, ISBN 0321194330
- www.w3c.org, Web Content Accessibility Guidelines 1.0 en 2.0

Sogeti Nederland B.V.

Het standpunt van Sogeti Nederland B.V. (Sogeti) is dat we willen bijdragen aan het resultaat van onze klanten door gepassioneerd ICT-vakmanschap. De visie van Sogeti valt daarom als volgt samen te vatten: ‘Resultaat door gepassioneerd ICT-vakmanschap’.

Dit betekent dat Sogeti verantwoordelijkheid neemt voor haar activiteiten bij klanten. Ze verplicht zich aan het resultaat op basis van haar excellente professionaliteit en haar ondernemerschap. Door hechte en langdurige relaties met klanten zet Sogeti ICT dusdanig in, dat het bijdraagt aan de strategische doelstellingen van haar klanten.

Binnen Sogeti Nederland B.V. is de divisie Software Control als eigenaar van de TMap®-methode en het TPI®-model en met haar 725 specialisten een trendsetter op het gebied van testen en quality assurance binnen het ICTwerkveld. De dienstverlening van Software Control concentreert zich rond requirements lifecycle management, gestructureerd testen, quality assurance en (test)proces verbetering. De vorm van deze dienstverlening varieert van detachering via testprojecten tot volledige uitbesteding van het testproces aan de TMap®Factory, al of niet met inschakeling van ons zusterbedrijf Sogeti India. De opdrachtgevers van Software Control zijn te vinden in alle segmenten van het Nederlandse bedrijfsleven en de overheid.

Om met haar dienstverlening voortdurend de ontwikkelingen en trends in de ICT-wereld te volgen, investeert Software Control veel in research & development. Innovaties in de technologie, nieuwe ontwikkelmethoden, nieuwe toepassinggebieden en trendswijzigingen in ICT-beleid van toonaangevende ondernemingen worden op de voet gevolgd.

De resultaten van research & development worden gepubliceerd in de vakbladen en in (internationale) newsletters en gepresenteerd op TMap® Test Topics seminars en de nationale en internationale (test)platforms als Testnet, SPIder en Eurostar.

Ook op www.tmap.net worden de resultaten in detail gepubliceerd.

Software Control is gecertificeerd conform ISO 9001:2000

Sogeti Nederland B.V.,
Lange Dreef 17
4131 NJ Vianen
www.sogeti.nl
www.tmap.net

Alfabetisch trefwoordenregister

A B C D E F G H I K L M N O P Q R S T U V W Z

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z