



SOGETI

Business  
driven test  
management

# Workbook TMap® Suite

*Studeren voor de TMap® Suite Certificering*



TMap®  
Suite



# Inhoud

<i>Inhoud</i> .....	2
<i>Inleiding</i> .....	6
<i>Hoofdstuk 1 Mr. Mikkel's overpeinzingen</i> .....	7
1.1 <i>Mr. Mikkel's overpeinzingen (1) over Building blocks</i> .....	7
1.2 <i>Mr. Mikkel's Overpeinzingen (2) over de elementen</i> .....	8
1.3 <i>Mr. Mikkel's overpeinzingen (3) over ingebouwde kwaliteit in een kwaliteitsgestuurde aanpak</i> .....	10
<i>Hoofdstuk 2 Building blocks</i> .....	14
2.1 <i>Building block 1: Testmanager</i> .....	14
2.2 <i>Building block 2: Testmanager in traditionele situaties</i> .....	14
2.3 <i>Building block 3: Opdracht</i> .....	15
2.4 <i>Building block 4: Testorganisatie</i> .....	16
2.5 <i>Building block 5: Testplan</i> .....	18
2.6 <i>Building block 6: Productrisicoanalyse</i> .....	19
2.7 <i>Building block 7: Teststrategie</i> .....	21
2.8 <i>Building block 8: Performance testen</i> .....	22
2.9 <i>Building block 9: Test approaches</i> .....	25
2.10 <i>Building block 10: Crowd testen</i> .....	26
2.11 <i>Building block 11: Testvariëteiten</i> .....	27
2.12 <i>Building block 12: Testmanager in Agile omgevingen</i> .....	28
2.13 <i>Building block 13: Permanente testorganisatie</i> .....	29
2.14 <i>Building block 14: Model based testen</i> .....	30
2.15 <i>Building block 15: Kwaliteitsbeleid</i> .....	32
2.16 <i>Building block 16: Gebruik van testtools</i> .....	35
2.17 <i>Building block 17: Kenmerken kwaliteitsgestuurde aanpak</i> .....	37
2.18 <i>Building block 18: Geïntegreerde testorganisatie</i> .....	39
2.19 <i>Building block 19: Implementatie testtools</i> .....	41
2.20 <i>Building block 20: Reviewen van requirements</i> .....	42
<i>Hoofdstuk 3 Website</i> .....	45
3.1 <i>Inleiding</i> .....	45
3.1.1. <i>Leeswijzer</i> .....	45
3.1.2. <i>Waarom testontwerp?</i> .....	45
3.1.3. <i>Voordeelen van testontwerp volgens de TMap Suite</i> .....	47
3.2 <i>Kader en begrippen testontwerp</i> .....	48
3.2.1. <i>Inleiding</i> .....	48
3.2.2. <i>De generieke testontwerpstappen</i> .....	49

3.2.3.	Dekking, dekkingsvormen en testzwaarte.....	52
3.2.4.	Test approaches .....	58
3.2.5.	Testontwerptechnieken.....	58
3.2.6.	Selectie van dekkingsvormen en testontwerptechnieken.....	59
<b>3.3</b>	<b>Dekkingsvormen Proces</b> .....	<b>60</b>
3.3.1.	Inleiding .....	60
3.3.2.	Paden .....	60
<b>3.4</b>	<b>Dekkingsvormen Condities</b> .....	<b>63</b>
3.4.1.	Inleiding .....	63
3.4.2.	Beslispunten .....	63
3.4.3.	Semantiek .....	73
<b>3.5</b>	<b>Dekkingsvormen Gegevens</b> .....	<b>73</b>
3.5.1.	Inleiding .....	73
3.5.2.	Equivalentieklassen .....	73
3.5.3.	Grenswaardenanalyse.....	75
3.5.4.	Gegevenscombinaties.....	77
3.5.5.	Syntax.....	79
3.5.6.	CRUD .....	79
3.5.7.	Integriteitsregels .....	82
<b>3.6</b>	<b>Dekkingsvormen Voorkomen</b> .....	<b>83</b>
3.6.1.	Inleiding .....	83
3.6.2.	Presentatie .....	83
3.6.3.	Load Profiles.....	84
3.6.4.	Operational Profiles.....	85
3.6.5.	Heuristiek .....	86
<b>3.7</b>	<b>Een basisset testontwerptechnieken</b> .....	<b>87</b>
3.7.1.	Inleiding .....	87
3.7.2.	Datacombinatietest (DCT) .....	87
3.7.3.	Procescyclustest (PCT).....	92
3.7.4.	Syntactische Test (SYN) .....	100
3.7.5.	Semantische Test (SEM) .....	102
3.7.6.	Beslistabeltest (BTT) .....	105
3.7.7.	Elementaire Vergelijkingentest (EVT) .....	112
3.7.8.	Gegevenscyclustest (GCT) .....	126
3.7.9.	Real Life Test (RLT) .....	130
<b>Hoofdstuk 4</b>	<b>TMap NEXT info</b> .....	<b>135</b>
<b>4.1</b>	<b>Wat is testen?</b> .....	<b>135</b>
4.1.1.	Wat is gestructureerd testen? .....	137
4.1.2.	Plaats van het testen .....	138
4.1.3.	TMap NEXT® in essenties .....	144
4.1.4.	Testen in een agile omgeving.....	156
4.1.5.	Proces acceptatie en systeemtesten .....	167
4.1.6.	Proces ontwikkeltesten .....	171
<b>4.2</b>	<b>Testprofessionals</b> .....	<b>172</b>
4.2.1.	Inleiding .....	172
4.2.2.	Aandachtspunten .....	172
4.2.3.	Eigenschappen.....	174
<b>4.3</b>	<b>Acceptatie en systeemtesten</b> .....	<b>175</b>

4.3.1.	Inleiding .....	175
4.3.2.	Fase Planning .....	178
4.3.3.	Fase Beheer .....	232
4.3.4.	Fase Inrichting en beheer infrastructuur .....	252
4.3.5.	4.3.5 Fase Voorbereiding .....	265
4.3.6.	Fase Specificatie.....	274
4.3.7.	Fase Uitvoering.....	288
4.3.8.	Fase Afronding .....	298
<b>4.4</b>	<b>Kwaliteitsattributen</b>	<b>301</b>
<b>4.5</b>	<b>Testomgeving</b>	<b>311</b>
4.5.1.	Inleiding .....	311
4.5.2.	Inrichting van testomgevingen.....	312
4.5.3.	Problemen bij testomgevingen .....	316
4.5.4.	OTAP-model.....	316
<b>4.6</b>	<b>Testtools</b>	<b>319</b>
4.6.1.	Inleiding .....	319
4.6.2.	Testtools toegelicht .....	320
4.6.3.	Soorten testtools.....	321
4.6.4.	Tools voor het vormgeven van de testomgeving.....	328
4.6.5.	Invoeren van testtools met toolbeleid .....	329
4.6.6.	Gewenste effecten, commitment, voorwaarden.....	330
4.6.7.	Implementatiefase .....	332
4.6.8.	Gebruik .....	337
<b>4.7</b>	<b>Bevindingen</b>	<b>338</b>
4.7.1.	Inleiding .....	338
4.7.2.	Een bevinding doen .....	339
4.7.3.	Bevindingrapport .....	344
4.7.4.	Procedure .....	348
<b>4.8</b>	<b>Ontwikkeltesten</b>	<b>350</b>
4.8.1.	Inleiding .....	350
4.8.2.	Ontwikkeltesten toegelicht.....	351
4.8.3.	Unittest .....	356
4.8.4.	Unitintegratietest.....	357
<b>4.9</b>	<b>Testbegroting opstellen</b>	<b>359</b>
4.9.1.	Begrotingstechnieken.....	359
4.9.2.	Begroten.....	359
4.9.3.	Begroten op basis van verhoudingsgetallen .....	362
4.9.4.	Begroten op basis van testobject omvang .....	363
4.9.5.	Work Breakdown Structure .....	365
4.9.6.	Toetsbegrotingsaanpak .....	365
4.9.7.	Proportioneel begroten.....	366
4.9.8.	Extrapolatie .....	366
4.9.9.	Testpuntanalyse .....	367
4.9.10.	Invoer en startvoorwaarden .....	371
4.9.11.	Punten voor testactiviteiten.....	372
4.9.12.	Toetsbare testpunten .....	377
4.9.13.	Totaal aantal testpunten.....	378
4.9.14.	Primaire testuren.....	378
4.9.15.	Totaal aantal testuren.....	381
4.9.16.	Verdeling over de fasen .....	383

4.9.17.	TPA in een vroegtijdig stadium .....	384
<b>4.10</b>	<b>Metrieken</b>	<b>384</b>
4.10.1.	Inleiding.....	384
4.10.2.	GQM-Methode in zes stappen.....	385
4.10.3.	Hints en Tips .....	388
4.10.4.	Praktische beginset testmetrics.....	388
4.10.5.	Metricslijst .....	391
<b>4.11</b>	<b>Toetstechnieken</b>	<b>392</b>
4.11.1.	Inleiding.....	392
4.11.2.	Toetsen toegelicht .....	392
4.11.3.	Inspecties.....	396
4.11.4.	Reviews .....	399
4.11.5.	Walkthroughs .....	400
4.11.6.	Keuzematrix toetstechnieken .....	402
<b>4.12</b>	<b>Kwaliteitsmaatregelen in de ontwikkelfase.</b>	<b>403</b>
<b>4.13</b>	<b>Beschrijven mastertestplan</b>	<b>412</b>
4.13.1.	Inleiding.....	412
<b>Literatuur</b>		<b>414</b>

*Afbeeldingen in dit document zijn gemaakt door Sogeti Nederland B.V. en mogen niet zonder wederhoor en duidelijke referentie per plaatje gebruikt worden in ander materiaal.*

# Inleiding

## Waarom dit workbook?

Sinds de introductie in 1995 is TMap (Test Management Approach) uitgegroeid tot de standaard voor het gestructureerd testen van software. Deze positie werd verder versterkt met de komst van TMap NEXT® in 2006. Testmanagers en Testers valideren hun professionaliteit door het behalen van het EXIN certificaat TMap NEXT® Test Master of Test Engineer.

De continue verbetering van de methode heeft in 2014 geleid tot de ontwikkeling van de TMap Suite. De TMap Suite bestaat uit de volgende onderdelen:

- De nieuwe aanpak: **TMap HD - Human Driven**. Een kwaliteitsgestuurde testaanpak voor de moderne, agile organisaties. Dit wordt beschreven in de roman "Neil's Quest for Quality".
- De nieuwe **TMap.net website**. Deze nieuwe website bevat de bouwstenen van TMap en kunnen worden gebruikt om een eigen testmethode te bouwen.
- **TMap NEXT®** is een testmethode voor organisaties die de traditionele ontwikkelmethoden zoals waterval gebruiken.



Dit workbook is ontwikkeld als ondersteuning voor het behalen van het EXIN certificaat TMap® Suite Test Manager en Test Engineer. Het is een verzameling van verschillende literatuurbronnen (vanuit TMap HD, de website en TMap NEXT) die gezamenlijk de leerstof voor de examens bepalen. Het is dus geen nieuw TMap boek en bevat geen nieuwe informatie ten opzichte van eerdere boeken. Uit de volgorde van de verschillende bronnen in dit boek kan niets worden afgeleid over het belang van het onderwerp of de volgorde waarin dit geleerd zou moeten worden. Het is slechts bedoeld als hulpmiddel voor examenkandidaten en om duidelijk te maken welke onderdelen van de TMap Suite tot de stof voor het examen behoren. Voor de samenhang tussen de verschillende onderwerpen kan men het beste de originele boeken raadplegen of een cursus die opleid voor deze certificering volgen.

In het eerste hoofdstuk komt TMap HD aan bod. De relevante onderdelen uit het boek in romanform zijn hierin opgenomen. Het tweede hoofdstuk gaat in op de bouwstenen (building blocks) van TMap. In het derde hoofdstuk wordt ingegaan op de website TMap.net, waarna in het vierde en laatste hoofdstuk de relevante informatie uit het TMap Next boek is opgenomen.

# Hoofdstuk 1 Mr. Mikkel's overpeinzingen

De onderstaande overpeinzingen zijn overgenomen uit het boek "Neil's quest for quality".

## 1.1 Mr. Mikkel's overpeinzingen (1) over Building blocks

"Een reis van duizend mijl begint met één stap."

Lao Tze

Over het algemeen, als ik iemand coach op het gebied van kwaliteit en testen, merk ik dat mensen al snel overweldigd raken als ze in één keer de hele methode voor testen en kwaliteit gepresenteerd krijgen. Het is vele malen eenvoudiger om elk onderdeel van de methode afzonderlijk te presenteren en ze bekend te laten worden met het ene deel voordat het volgende geïntroduceerd wordt. Het is vaak het handigst om degenen die worden gecoacht te laten starten met de onderdelen die voor hun situatie het belangrijkste zijn. Als ze dat hebben geleerd en geïmplementeerd, begin dan pas aan het volgende onderdeel.

Hetzelfde geldt op een grotere schaal. Als organisaties hun test- en kwaliteitsmethode willen aanpassen, is het voor hen ook gemakkelijker om een deel goed te leren kennen, dit binnen hun organisatie te implementeren om zo een specifiek probleem op te lossen en vervolgens de volgende uitdaging te lijf te gaan. Een organisatie die in één keer met een heel nieuw proces wordt geconfronteerd zal dit minder goed begrijpen. In die gevallen zullen veel processtappen en tools niet goed worden begrepen en dus ook niet goed worden gebruikt. Dit leidt tot een situatie waar het volgen van een proces het doel wordt in plaats van het oplossen van het probleem waar het proces voor bedoeld is. Dit leidt tot 'in-name-only' varianten van standaard methoden.

Elke organisatie is verschillend en stelt andere eisen aan zijn testmethode. Ben je Agile, of meer traditioneel? Ben je verplicht om te conformateren aan formele kwaliteitsstandaarden of niet? Heb je zeer ervaren mensen in de organisatie of ben je een jong en gretig bedrijf waar nog veel moet worden geleerd?

Al deze dingen en meer hebben een invloed op de methode voor kwaliteit en testen die het beste bij jouw organisatie past. Dit betekent dat elke organisatie zijn eigen optimale methode heeft. Het is zelfs zo dat een methode optimaal kan zijn voor een organisatie op een bepaald moment – maar minder optimaal op een ander moment, als er bepaalde organisatiwijzigingen hebben plaatsgevonden. Bijvoorbeeld, de introductie van een nieuwe tool die het eenvoudiger maakt om bepaalde zaken te testen kan een aanpassing aan de testmethode vereisen.

Het helpt mensen en organisaties als ze een methode stap voor stap op kunnen bouwen en ontwikkelen – met behulp van Building blocks. Een Building block is een processtap, tool of een rol die een bepaald test- en kwaliteitsprobleem binnen jouw organisatie kan oplossen. Een Building block kan ook in de bestaande methode worden ingepast, of verplaatst binnen de methode. Bijvoorbeeld, een specifieke test kan eerder in het ontwikkelproces worden uitgevoerd om te zorgen dat een bepaald soort fouten eerder gedetecteerd wordt.

Je kunt ook je eigen Building blocks ontwikkelen. Als jouw organisatie bijvoorbeeld aan specifieke standaarden moet voldoen kun je een speciale Building block maken om te controleren of aan deze standaarden wordt voldaan.

Een goed startpunt is de verzameling Building blocks die te vinden is op [www.tmap.net](http://www.tmap.net). Voel je vrij om deze te gebruiken en aan te passen aan jouw specifieke situatie.

Ik kan me voorstellen dat je meer inspiratie kan gebruiken over hoe je meerdere Building blocks combineert. Dit is ook een onderdeel van de TMap Suite en kan ook op tmap.net worden gevonden.

## 1.2 Mr. Mikkel's Overpeinzingen (2) over de elementen

*"Het is elementair, mijn beste Watson."*  
Sherlock Holmes, in de film 'De terugkeer van Sherlock Holmes'

Toen ik Neil coachte, liet ik hem kennismaken met de vijf elementen van kwaliteitsgestuurd testen.

Deze elementen hebben twee doelen. Aan de ene kant zijn het elementen van de ontwikkeling van het kwaliteits- en testvak. Het vakgebied van kwaliteit en testen is aan het veranderen en deze elementen geven de richting van de verandering aan. De elementen helpen Neil ook om keuzes te maken, om betere resultaten te bereiken en om antwoord te vinden op vragen als:

Wat is de beste teststrategie?

- Hoe kan ik meer en sneller testen?
- Hoe kan ik een betere kwaliteit bereiken?
- Welke building blocks kan ik gebruiken?
- Hoe kan ik die bouwbladen toepassen?

### Vereenvoudig (Simplify)

*"Alles moet zo eenvoudig mogelijk gemaakt worden. Maar niet eenvoudiger."*  
Albert Einstein

Sinds het begin van IT in het bedrijfsleven is het IT landschap steeds complexer geworden. Het gevolg voor testen en kwaliteit is dat door groeiende complexiteit er meer testen noodzakelijk is om alle relaties en effecten daarvan op de keten van IT oplossingen te adresseren. Om deze opgaande spiraal te beëindigen, is het van belang te vereenvoudigen, te standaardiseren en te ontkoppelen. Het test- en kwaliteitsvakgebied kan haar stappen vereenvoudigen in lijn met de vereenvoudigingen van de IT-omgeving. Afgezien van de vereenvoudiging van testen in lijn met de vereenvoudiging van het IT-landschap, kan de efficiëntie van testactiviteiten worden verbeterd door de activiteiten kleinschalig en duidelijk te houden: alleen die testactiviteiten die nodig zijn om klantwaarde te bereiken worden uitgevoerd. Teststrategie en testtechnieken moeten worden gekozen op een manier die het beste past bij een specifieke situatie.

### Integreer (Integrate)

*"Elke vorm van vreedzame samenwerking onder de mensen is voornamelijk gebaseerd op wederzijds vertrouwen en pas in tweede instantie op instellingen zoals rechtkanten en politie."*

Albert Einstein

Deel van de IT-evolutie is de noodzaak te integreren. IT complexiteit wordt verminderd met het structureren, vereenvoudigen en standaardiseren van IT oplossingen binnen een samenhangend IT landschap, en door het integreren van IT oplossingen met bedrijfsprocessen.

Het proces van creëren van dergelijke oplossingen staat onder enorme druk. Integratie is een van de antwoorden, waar alle disciplines die betrokken zijn bij het proces van het

creëren van IT-oplossingen beter moeten samenwerken om de efficiëntie, snelheid en kwaliteit te verhogen.

Met integratie op het gebied van testen wordt een gedeelde manier van werken bedoeld, met een gedeelde verantwoordelijkheid voor de kwaliteit. Testen is geen zelfstandig proces en zou naadloos moeten aansluiten op de omringende processen. De integratie van testen en kwaliteitsbenaderingen is niet nieuw. Testen is één van de maatregelen om een risico te beheersen. Soms kan een risico worden beheerd door zwaarder testen, soms is het beter om het te beheersen door andere kwaliteitsmaatregelen zoals pair-programming of testgedreven technieken. De essentie hierbij is dat in een geïntegreerde aanpak een risico niet hoeft te worden afgedekt door testen.

### **Industrialiseer (Industrialize)**

*“De monotonie van een rustig leven stimuleert de creatieve geest.”*

*Albert Einstein*

De standaardisatie van testen biedt mogelijkheden voor de automatisering van testuitvoering. Modellen kunnen worden gebruikt voor het automatisch genereren van testgevallen. In feite kan elk soort testactiviteit door tools worden ondersteund. Zo kan bijvoorbeeld de planning van een test worden ondersteund door testmanagement tooling, de specificatie van een test door op modellen gebaseerde methoden, zijn er testuitvoeringstools, en kan een testomgeving worden beheerd met service virtualisatie en testdata-management tooling. Er zijn ook geïntegreerde tools, kwaliteitssuites of levenscyclussuites.

Industrialiseren is zeer belangrijk bij het optimaliseren van testen en verbeteren van kwaliteit.

Testtools kunnen worden gebruikt voor vaker, meer en sneller testen. Meer informatie over industrialiseren kan worden gevonden in de Building blocks die over testtools en het gebruik ervan gaan.

Industrialiseren impliceert aspecten zoals:

- Testautomatisering;
- Acceleratoren;
- Standaardisatie;
- Hergebruik;
- Testontwerptechnieken;
- Templates;
- Testomgevingen.

### **Mensen (People)**

*“Het belangrijkste is om nooit op te houden met vragen stellen.”*

*Albert Einstein*

Een methode is één, het toepassen van die methode is iets anders. Verschillende project management methodes, verschillende bedrijfsculturen, verschillende kwaliteitseisen, verschillende omgevingen: ze vragen allemaal om een wijze toepassing van elke methode. Zonder de juiste mensen om een methode uit te voeren, zal elke methode falen.

Mensen moeten de juiste vaardigheden en kennis bezitten om hun werk uit te voeren. In een kwaliteitgestuurde aanpak is de juiste mindset voor ‘First time right’ ook essentieel.

Iedereen heeft een bepaald beeld bij kwaliteit. Kwaliteitsprofessionals zijn getraind om relevante kwaliteitsaspecten tastbaar en meetbaar te maken.

Iedereen in een organisatie kan tests uitvoeren, als ze geholpen worden door professionele testers met de kritische instelling om adequaat en effectief te testen.

Het element De Mens maakt het mogelijk om te groeien van testen volgens TMAP naar testen mét TMAP. Hiervoor zijn mensen nodig met een brede kennis van kwaliteit en testen, met de juiste instelling om de bouwblokken toe te passen op een manier die past bij hun organisatie. Vandaar de naam TMAP HD: human driven (mensgedreven).

### **Vertrouwen (Confidence)**

*"Niet alles wat geteld kan worden telt, en niet alles dat telt kan geteld worden."*  
Albert Einstein

Een vijfde element komt voort uit de vier overige elementen: Vertrouwen. Dat is een extra element boven de anderen. De vier elementen verbeteren de testaanpak. Gezamenlijk vormen ze de essentiële basis waar vanuit het vijfde element ontstaat: Vertrouwen is waar ze allen naar leiden. Het is dit vijfde element dat een kwaliteitsgestuurde aanpak onvermijdelijk maakt. De noodzaak voor betrouwbare IT oplossingen neemt toe als de afhankelijkheid van IT toeneemt.

Kwaliteit wordt vaak gedefinieerd als 'geschikt voor gebruik', maar sommigen in mijn omgeving stellen dat kwaliteit een irrationeel gevoel is, en daarom niet gedefinieerd kan worden. Feit is dat deze vier elementen onmisbaar zijn in het creëren van vertrouwen in IT oplossingen.

## **1.3 Mr. Mikkel's overpeinzingen (3) over ingebouwde kwaliteit in een kwaliteitsgestuurde aanpak**

### **Perspectieven**

Mensen kijken normaal gesproken naar een situatie vanuit hun eigen gezichtspunt, gebaseerd op hun positie, ervaringen en waarden. Dezelfde situatie kan worden beschreven vanuit deze gezichtspunten, resulterend in verschillende verhalen, soms identiek, soms schijnbaar in tegenspraak met elkaar (zie figuur 1).



Figuur 1. Verschillend inzicht door verschil van perspectief

Seabiscuit is een project als vele anderen. Neil beschreef het vanuit zijn gezichtspunt, vanuit het perspectief van kwaliteit- en testmanager. Dit perspectief was nogal nieuw voor hem en ik werd gevraagd om hem te begeleiden. Owen, Rupert, Francine, Rajiv, Hal en zeker Danielle hadden andere invalshoeken (zie "Neil's Quest for Quality" voor rol van de projectleden). Het is belangrijk om te beseffen dat iedereen naar dezelfde werkelijkheid keek.

### Kwaliteit wordt ingebouwd in het proces

Rupert's interventie, waarmee hij een kwaliteitsgestuurde aanpak eiste om het vertrouwen in IT-oplossingen te verbeteren, zorgde voor een verandering in Neils perspectief vanaf het einde van het ontwikkelingsproces naar een coördinerende positie. In een kwaliteitsgestuurde aanpak is het essentieel dat de kwaliteit wordt ingebouwd in het proces. Tests worden gebruikt om de kwaliteit gedurende het gehele proces te monitoren. Ingebouwde kwaliteit is een van de belangrijkste principes in de Lean aanpak, evenals continue verbetering, elimineren van afval(waste) en het waarderen van mensen. Ingebouwde kwaliteit, die voortdurend wordt verbeterd, leidt tot 'in één keer goed', waarbij de uitkomst van het proces volledig voldoet aan de verwachting: geschikt voor gebruik.

### Afhandelen van test- en kwaliteitsissues

Kwaliteit heeft in deze context betrekking op de kwaliteit van het resultaat van het proces: de productkwaliteit. Het proces en de kwaliteit van het product zijn sterk met elkaar verbonden. Klantwaarde is hierbij een essentiële factor en dat is waarom een kwaliteitsgestuurde aanpak ook businessgedreven moet zijn. Een product wordt gespecificeerd en ontworpen voor alle fasen van de levenscyclus. Elke afwijking van de verwachte productkwaliteit dient zo snel mogelijk gedetecteerd te worden en moet leiden tot maatregelen. Herstellen van de fout is niet genoeg, het is essentieel om het proces te verbeteren zodat dergelijke fouten niet opnieuw ontstaan. Zo wordt kwaliteit ingebouwd. Zo wordt de productkwaliteit verbeterd door aanpassing van het proces. Een extra test of een test verbeteren (bijv. regressietest) kan één van dergelijke aanpassingen zijn, naast andere kwaliteitsmaatregelen. Dat is ook de reden waarom Neil behoefte had aan een manier om het hele proces te overzien en aan een manier om de totale aanpak te beïnvloeden, zelfs voorafgaand aan de feitelijke start, door de projectleider en de teams te ondersteunen. Testen is geïntegreerd in het ontwikkelingsproces.

## **De elementen gebruiken voor een kwaliteitsgestuurde aanpak**

Vanuit zijn kwaliteits- en testperspectief, gebruikte Neil de beschikbare Building blocks op een verbeterde wijze ('Fit to purpose'). Ik heb de elementen om dit te doen genoemd: Integreer, De Mens, Industrialiseer en Vereenvoudig, waarbij het testen leidt tot Vertrouwen in de ontwikkelde oplossingen.

Vereenvoudig is altijd belangrijk in alle activiteiten, en moet worden gedaan wanneer de kans daartoe bestaat. Vereenvoudiging van de aanpak door het creëren van korte cycli om veranderingen klein en eenvoudig te houden is een voorbeeld.

- *Integreer* is belangrijk om samen te werken, korte communicatielijnen te hebben en te werken op een businessgedreven manier. Kwaliteit is een gedeelde verantwoordelijkheid van iedereen die zich bezighoudt met de ontwikkeling van het product.
- *Industrialiseer* is belangrijk om vele controles tijdens het gehele proces in te bouwen, om afwijkingen automatisch te detecteren. In Lean manufacturing worden deze controles zoveel mogelijk geautomatiseerd.
- *De Mens*: de houding en mentaliteit van mensen leveren een grote bijdrage aan de kwaliteitsgestuurde aanpak. Als mensen verantwoordelijk worden gehouden voor de kwantiteit van hun productie, of voor het leveren voor een deadline zonder een gedefinieerde kwaliteitsnorm, moet men niet verbaasd zijn als de geleverde kwaliteit laag is. Als je mensen verantwoordelijk maakt voor een hoog kwaliteitsniveau en ze laat beslissen over hun werkproces, kan een andere uitkomst worden verwacht.
- *Vertrouwen* is de belangrijkste motivatie voor een kwaliteitsgestuurde aanpak.

## **Het ontwikkelingsproces: Agile - Waterval**

Welk ontwikkelingsproces daadwerkelijk wordt gebruikt is van minder belang.

Kwaliteitsmaatregelen kunnen worden ingebouwd in elk proces. In het eerste deel van het Seabiscuit project, gebruikte ZBO een waterval proces; in het tweede deel een Agile proces. Niet omdat waterval slecht is en Agile beter. Kwaliteitsgestuurde elementen (vereenvoudig, integreer, industrialiseer, mensen) kunnen worden toegepast in alle ontwikkelmethodes.

Agile heeft een aantal Lean principes ingebouwd, maar het is de kwaliteitsgestuurde mentaliteit van de mensen dat het succesvol maakt. Echter, Agile werken is niet altijd succesvol en niet altijd geschikt of van toepassing, terwijl watervalprojecten zeer succesvol kunnen zijn.

## **Effect op lange termijn: project versus staforganisatie**

Essentie is de mentaliteit gericht op kwaliteit door middel van continue verbetering. Het streven is nul fouten, dat betekent dat er geen afwijkingen zijn ten opzichte van de gespecificeerde kwaliteitscriteria. Lean manufacturing heeft laten zien dat deze continue verbeteringen na verloop van tijd zullen leiden tot stijgende kwaliteit en dalende kosten. Er wordt een keuze gemaakt ten gunste van lange termijn effecten. Projecten zijn tijdelijke organisatiestructuren die niet altijd geschikt zijn om lange termijn effecten te verkrijgen. Daarom is een permanente, projectoverkoepelende organisatie belangrijk: het bestaat uit gekwalificeerd personeel, die werken aan een kwaliteitsbeleid, testdeskundigheid, beleid ten behoeve van testtools etc. Dit beleid wordt bij de start van projecten aan hen doorgegeven.

## **Conclusies**

Dus waterval of Agile is niet een keuze tussen goed of slecht, maar een oproep om beide zorgvuldig te overwegen. Het zal altijd een kwestie van afweging zijn. Er zijn ook situaties waarin een quick and dirty wegwerplossing de beste keuze is. Of de beste start, waarbij de wegwerplossing opgevolgd wordt door een lange termijn eindoplossing, met lagere uitvoeringskosten bijvoorbeeld, of ondersteuning biedend aan een effectiever bedrijfsproces. Het hangt allemaal af van het beoogde doel.

Het is die geschiktheid voor het doel, ook wel 'kwaliteit' genoemd, die de kwaliteits- en testaanpak bepaalt.

Projectvorm, ontwikkelmethode, teststrategie, etc.: deze zijn allemaal aan elkaar verbonden en moeten worden geïntegreerd. Een adequate geïntegreerde methode is toepasbaar voor elk scenario.

Een testaanpak wordt bepaald door de Building blocks op een passende wijze toe te passen.

Ik ben ervan overtuigd dat geschikte patronen van Building blocks, nieuwe methodes en best practices voor veel diverse specifieke situaties zullen ontstaan. Als we allemaal samenwerken en delen, hebben we daar allemaal voordeel aan.

## **Hoofdstuk 2 Building blocks**

Deze Building blocks komen uit het boek "Neil's quest for quality".

### **2.1 Building block 1: Testmanager**

Wat doen testmanagers? In traditionele organisaties wijzen ze mensen toe aan projecten, houden toezicht op de voortgang van de testers, geven feedback en bieden misschien wat coaching aan mensen die dat willen. Testmanagers bouwen vertrouwensrelaties op met hun personeel en werken aan de opbouw van de capaciteit van de testgroep. Hoe gaat dat veranderen met een overgang naar Agile? Is er nog behoefte aan testmanagers? De antwoorden op deze vragen worden gegeven in de Building blocks 'Testmanager in traditionele omgevingen' en 'Testmanager in agile omgevingen'. De eerste zal direct onder deze Building block worden gegeven, en de 'Testmanager in Agile omgevingen' Building block vindt men in Building block 12.

In dit boek maken we gebruik van 'testmanager' als een generieke term. In de praktijk kunt u veel verschillende termen tegenkomen die verwijzen naar deze rol, zoals 'testcoördinator', 'testleider', 'projectleider testen', 'testdirecteur', en nog veel meer. Soms verwijzen deze termen naar verschillende niveaus in de organisatie, wanneer bijvoorbeeld meerdere testcoördinatoren ondergeschikt zijn aan een testmanager. Wij raden u altijd aan een duidelijke definitie van de rol en de verantwoordelijkheden in uw specifieke situatie te maken.

### **2.2 Building block 2: Testmanager in traditionele situaties**

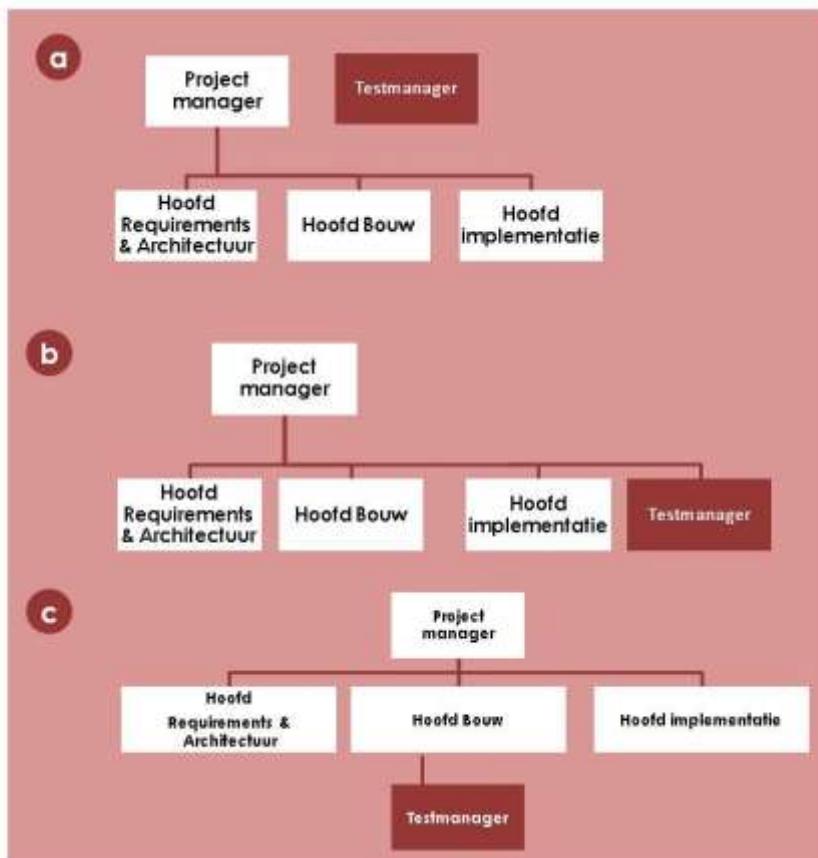
In traditionele organisaties geeft de testmanager leiding aan een team van testcoördinatoren en/of testers. Omdat de testmanager het gehele testproces overziet, moet hij in staat zijn om een versnipperde aanpak te voorkomen. De testmanager van vandaag tracht ook de focus te verleggen van testen aan het einde van een project naar andere kwaliteitsmetingen die bij het begin van een project kunnen worden uitgevoerd, zoals reviews, inspecties en conceptvalideringen. Hij of zij is de centrale persoon bij het opstellen van de teststrategie, waarbij alle benodigde partijen en informatie samengebracht worden. De testmanager is verantwoordelijk voor de planning, het beheer en de uitvoering van het testen, ervoor te zorgen dat het op tijd, binnen budget en met de juiste kwaliteit plaatsvindt, over meerdere testvariëteiten heen. De testmanager rapporteert conform het master testplan over de voortgang van het testproces en de kwaliteit van het testobject.

Voorbeelden van taken van de testmanager:

- Het creëren van de instructies voor de testproducten die de verschillende testvariëteiten opleveren.
- Het controleren van naleving van de instructies (interne reviews).
- Het coördineren van de verschillende testactiviteiten die van toepassing zijn op de testvariëteiten, zoals het opzetten en beheren van de technische infrastructuur.
- Het creëren van richtlijnen voor de communicatie en rapportage tussen de testvariëteiten, het testproces en de leveranciers.
- Het opzetten van de totale testmethode gerelateerde, technische en functionele ondersteuning.
- Het consistent houden van de verschillende testplannen.
- Rapporteren over de algehele testvoortgang, budget en kwaliteit van het testobject, bij voorkeur geademtiseerd met een testmanagementtool.

- Managen van de verwachtingen van de verschillende belanghebbenden met betrekking tot de voortgang en de kwaliteit van het testen.
- Inzetten/inhuren van (extra) testpersonnel.

De relatie tussen de vastgestelde rollen, de testvariëteiten en de relaties met de andere belanghebbenden in het systeemontwikkelproces moet worden bepaald en gedocumenteerd. De testorganisatie maakt duidelijk deel uit van het grotere (project) plaatje. Zie figuur 2 voor enkele voorbeelden. In deze voorbeelden zijn rapportagelijnen en ondersteunende afdelingen, zoals bijvoorbeeld het testexpertisecentrum, weggelaten.



Figuur 2. Voorbeelden van posities van de testmanager in projecten

Toelichting bij de voorbeelden:

- Voorbeeld a: de testmanager is volledig onafhankelijk van zowel de projectmanager als de deelprojectleider realisatie, en werkt op hetzelfde niveau als de projectmanager.
- Voorbeeld b: de testmanager is afhankelijk van de projectmanager, maar onafhankelijk van de deelprojectleider realisatie, en werkt op hetzelfde niveau als de deelprojectleider realisatie.
- Voorbeeld c: de testmanager is afhankelijk van de deelprojectleider realisatie.

## 2.3 Building block 3: Opdracht

Over het algemeen geldt: geen klus zonder een opdracht. Dit geldt ook voor een klus zoals het vaststellen van de kwaliteit van een product. Op een hoog niveau zou men kunnen zeggen dat de opdracht aan de stakeholders precies duidelijk moet maken wat

het doel, de taken, verantwoordelijkheden en bevoegdheden van betrokkenen bij de klus zijn.

Opdrachten komen in verschillende variëteiten. Zo zal een opdracht in een traditionele ontwikkelomgeving verschillen van een opdracht in een Agile ontwikkelomgeving en een opdracht in een lage risico-omgeving zal verschillen van een opdracht in een risicovolle omgeving. Bepakt dit dat er geen algemene activiteiten zijn die kunnen worden ondernomen om een opdracht neer te zetten? Integendeel. De vorm van het eindresultaat van deze activiteiten - in een plan van vele pagina's of alleen een schets op een white board - kan variëren, maar de generieke activiteiten kunnen zeker worden geïdentificeerd.

Een **niet-uitputtende** lijst van mogelijke vragen die u moet beantwoorden om de opdracht te verduidelijken:

1. Wie is de opdrachtgever? De opdrachtgever is de persoon die de opdracht geeft voor de taak. Het zou een business manager kunnen zijn, maar ook een projectmanager, stuurgroep, scrum team, etc.
2. Wie is verantwoordelijk voor de uitvoering van de opdracht? Het zou een scrum team kunnen zijn, maar ook een testmanager, projectmanager, een derde, etc.
3. Wat is precies de taak? Denk aan het bepalen van de kwaliteit van het product, de (product) risico's, het bereiken van bepaalde vooraf vastgelegde testdoelen, etc.
4. Wie zijn de acceptanten? Vaak is de persoon die de opdracht uitvoert niet degene die het product accepteert. Dus wie wel? Het zou een business manager kunnen zijn, een product owner, een groep van belanghebbenden, een scrum team, beheer, etc. Let op: de klant hoeft niet de - enige - acceptant te zijn.
5. Wat zijn de acceptatiecriteria?
6. Bepaal de omvang van de opdracht. Bepaal niet alleen wat er binnen het kader valt, maar ook wat buiten het bestek van de opdracht valt. Bepaal bijvoorbeeld de grenzen van het systeem (de interfaces met aanpalende systemen zijn binnen/buiten scope), of de procedures van de administratieve organisatie binnen de scope vallen of niet, voor zover relevant welke testsoorten (bijvoorbeeld unit-test, systeemtest, acceptatietest) binnen scope vallen en welke worden uitgevoerd door andere partijen, etc.
7. Aan welke randvoorwaarden moet worden voldaan? Randvoorwaarden beschrijven de eisen die door andere partijen binnen de opdracht gesteld worden aan de taak. Dergelijke eisen kunnen zijn: 'opereren binnen het bestaande kwaliteits-, risico- en/of testbeleid', 'aspecten afdekken van eerder gekwantificeerde zaken zoals de af te dekken risico's, resultaten en te bereiken kwaliteit', de tijdslimiet in een business case of andere planningschema's, etc.
8. Welke informatie zullen we delen met de stakeholders? Dit kan op vele manieren. Afspraken worden vaak gemaakt met de opdrachtgever en eventuele andere belanghebbenden over rapportage. Vaak zijn risico's- en kwaliteitsrapportages gewenst. Rapportage kan worden gedaan op een periodieke of ad hoc basis en/of aan het einde van een project. Veel organisaties hebben standaard templates voor rapportages.

## 2.4 Building block 4: Testorganisatie

Het beheer en de uitvoering van testen kan in vele vormen worden geïmplementeerd. Immers, iedereen is verantwoordelijk voor de kwaliteit en bijna iedereen is betrokken bij één of andere vorm van testen. Daarom is het onmogelijk om te bepalen welke organisatorische vorm voor het testen de voorkeur heeft. In het algemeen moet de

structuur van de testorganisatie lijken op die van het bijbehorende systeemontwikkelproces. In veel gevallen betekent dit de projectorganisatie. Als er veelvuldig (her)testen in combinatie met schaarse (test)kennis benodigd is, kan een permanente testorganisatie (bijvoorbeeld staf- of lijnorganisatie) een goed alternatief zijn.

### Organisatievormen voor het testen

De belangrijkste organisatievorm worden hieronder kort vermeld, samen met enkele voorbeelden. Voor de organisatie van testactiviteiten kunnen de mogelijkheden in grote lijnen als volgt worden gedefinieerd:

- Testen als een **zelfstandige activiteit** of **geïntegreerd met andere activiteiten**
- Testen binnen een **project**, een **permanente (test)organisatie** of in de **cloud**

Deze keuzes zijn afhankelijk van de testvariëteit (zie Building block 11 ‘testvariëteiten’), het project en de organisatie. Voor mogelijke organisatievormen zie figuur 3.

Testen	Onafhankelijke activiteit	Geïntegreerd met andere activiteiten
Projectorganisatie	<ul style="list-style-type: none"> <li>• Systeemtest</li> <li>• Acceptatie test</li> <li>• Security, Performance, Usability test</li> </ul>	<ul style="list-style-type: none"> <li>• Unit test</li> <li>• Test in Agile omgeving (i.e. SCRUM, DevOps)</li> </ul>
Permanente organisatie	<ul style="list-style-type: none"> <li>• Testfabriek</li> <li>• Test line</li> <li>• Test expertise centrum</li> </ul>	<ul style="list-style-type: none"> <li>• Onderhoudsproces, end to end testorganisatie</li> </ul>
Cloud	<ul style="list-style-type: none"> <li>• Crowd test (i.e. beta, compatibiliteit, usability, game, mobilitätstest)</li> </ul>	(geen gewone voorbeelden)

Figuur 3. Organisatievormen met voorbeelden

De beschrijvingen hieronder zijn expliciet bedoeld als een generieke beschrijving; vaak zijn er uitzonderingen in de praktijk:

- **Testen als een zelfstandige activiteit in een project**

Binnen het project is een team verantwoordelijk voor het organiseren en uitvoeren van de test. De testers binnen het team hebben meestal veel testkennis, samen met - afhankelijk van de testsoort - een mix van systeem, domein en organisatorische kennis.

- **Testen geïntegreerd binnen een project**  
In de traditionele ontwikkelingsprojecten zijn testactiviteiten met unit testen geïntegreerd in het ontwikkelingsproces. In Agile projecten werken testers, gebruikers, ontwerpers en ontwikkelaars in hetzelfde team. Er zijn vaak meerdere teams (bijvoorbeeld scrum of DevOps teams) in actie. Testen is een rol binnen dit team en het team als geheel is verantwoordelijk voor het uitvoeren van de test. Naast diepgaande testkennis heeft elk teamlid met een testrol *als regel* ook veel domein- en technische kennis van het systeem en de architectuur (zie Building Block 18 'Geïntegreerde testorganisatie').
- **Testen als een onafhankelijke permanente organisatie**  
Een aparte afdeling of organisatie heeft testen - zowel de organisatie als de uitvoering - als haar primaire taak. Projecten of andere lijnafdelingen geven een zekere testinstructie naar deze afdeling / organisatie. Testkennis overheerst. (Zie Building block 13 'Permanente testorganisatie' voor twee veel voorkomende vormen van organisatie).
- **Testen geïntegreerd in de lijnorganisatie**  
Binnen een ontwikkelings- of systeembeheerafdeling, wordt de rol van tester vaak gecombineerd met andere functies (bijvoorbeeld, met de rol van functioneel beheerde). De tester in deze organisatievorm bezit vaak aanzienlijke systeemkennis en / of organisatorische kennis.
- **Testen als een zelfstandige activiteit in de cloud**  
Crowd sourced testen is een opkomende trend in het testen van software die gebruikt maakt van de voordelen, effectiviteit en efficiëntie van crowdsourcing en het cloud-platform. Het verschilt van de traditionele testmethoden doordat het testen wordt uitgevoerd door verschillende testers vanuit diverse plaatsen, en niet door ingehuurde consultants en professionals. Vaak zijn deze testers zeer bekwaam, maar de kwaliteit van het testen van het werk kan variëren, omdat je nooit weet wie betrokken zal worden bij een bepaalde testtaak.

## 2.5 Building block 5: Testplan

Een bekende uitspraak in het Engels is: "*Failing to plan is planning to fail*" : nalaten te plannen is plannen om te falen. Dus hebben we een testplan nodig - en wat zou daarin moeten staan? Projectplannen worden vaak gemaakt, maar regelmatig nauwelijks gelezen in de dagelijkse praktijk. Dus moet er een apart testplan zijn? Er zijn vragen over testen die opduiken in alle situaties, die moeten worden beantwoord.

Denk aan:

- Wat hebben we nodig om te testen?
- Wie zal wat testen op welk moment?
- Hoe zullen we testen?
- Hoeveel tijd zal het testen nemen?
- Wanneer is de test klaar?
- Hoe kunnen we het testen organiseren en beheren?
- Wat voor soort testproducten hebben we nodig om op te kunnen leveren en kunnen we ze beheren?
- Wat voor soort testomgeving hebben we nodig?

Natuurlijk kunnen er meer vragen zijn, maar dit zijn de belangrijkste.

Het uitwerken van deze thema's helpt ons over belangrijke zaken na te denken en dwingt ons om de uitdagingen die ons te wachten staan te confronteren. De resultaten van de

uitwerking (die bij voorkeur verschillende opties kennen) worden vastgesteld in een testplan, afhankelijk van de toegepaste ontwikkelmethode, (product)risico's, vertrouwen, doel, voorschriften, verantwoordelijkheden, etc. Omdat een plan vooral een middel is om te communiceren met de verschillende betrokken partijen, kunnen de volgende vragen worden gesteld:

- Waarom schrijven we dit in het plan: heeft het team dit nodig met het oog op het uitvoeren van ons werk?
- Voor wie schrijven we dit in het plan: moeten de stakeholders dit weten?

Het antwoord is natuurlijk situationeel afhankelijk, maar als het team en de belanghebbenden beide vragen negatief beantwoorden, lijkt het opschrijven van deze onderwerpen in het plan overbodig.

Dus, hoe kan een plan eruit zien? In de praktijk varieert dit van een schets op een whiteboard tot in projectplannen (of sprintplannen) geïntegreerde testaspecten, tot documenten met veel pagina's. Vaak zijn er Master Test Plannen (afstemmen van de verschillende testsoorten) en Detail Testplannen (uitgewerkt testplannen op testsoort niveau). En soms is er helemaal geen zichtbaar plan. Bijvoorbeeld wanneer men samenwerkt als een team en de leden elkaar blindelings begrijpen.

## 2.6 Building block 6: Productrisicoanalyse

De projecten moeten een integraal test- en kwaliteitsbeleid volgen, want kwaliteit is een mentaliteit, meer dan een eigenschap. Kwaliteit als zodanig moet een integraal onderdeel zijn van projectmanagement. Kijkend naar project governance, spelen een aantal aspecten een belangrijke rol: kosten, risico's, tijd en baten (ook wel aangeduid als 'resultaten' of 'klantwaarde').

Hoewel al deze aspecten van belang zijn, is het de moeite waard het risicoaspect te benadrukken, en dan met name het productrisico, omdat dit kan helpen als stuurmechanisme. Het kan u helpen om de balans te vinden tussen 'het bouwen van het juiste ding', 'het ding juist te bouwen', en 'het ding snel bouwen'.

Geen project heeft onbeperkte tijd, geld en middelen voor het beoordelen van de kwaliteit van het product. Dergelijke beperkingen in termen van tijd, geld en middelen vertegenwoordigen beperkingen op het te bereiken resultaat en betekenen daarom vaak beperkingen op de mogelijkheden om de productrisico's vast te stellen. Als zodanig is het belangrijk om een weloverwogen evenwicht te bereiken tussen de investering in tijd en geld enerzijds, en de te behalen resultaten en de beheerde risico's anderzijds. Het resultaat van de productrisicoanalyse biedt de rechtvaardiging voor dit evenwicht. Op basis van de inzichten uit de productrisicoanalyse, kunnen producten met een hoog risico intensiever worden getest dan producten met een lager risico. Wees ervan bewust dat de risico's en de manieren om deze af te dekken rechtstreeks verband houden met de acceptatiecriteria (zie Building block 3 'Opdracht'). Deze acceptatiecriteria zijn er in verschillende vormen: als een onderdeel van een testplan, in de bevestigingsmail, deel van een story card, in een definition of done, enz.

Risico's kunnen op diverse manieren worden bepaald. Wat alle benaderingen echter gemeen hebben is dat de productrisicoanalyse als doel heeft te komen tot een gedragen beeld - voor en met alle belanghebbenden - van hogere en lagere risiconiveaus van (de eigenschappen van) het te beoordelen product. Zodanig dat passende maatregelen kunnen worden genomen op basis van dat beeld.

Precies welke maatregelen moeten worden genomen om deze geanalyseerde risico's te beheersen wordt besloten bij het bepalen van de kwaliteitsstrategie. Dit betekent dat de juiste kwaliteit ingebouwd en ontworpen is, er niet erin getest wordt! Dit betekent dat iedereen risico- en kwaliteitsdenken vanaf het begin van het project moet omarmen. Op het moment dat een requirement wordt beschreven (bijvoorbeeld user story, use case) moet al nagedacht worden over mogelijke risico's en hoe deze te beheersen, bijvoorbeeld door het uitvoeren van een inspectie op de requirements (zoals een Fagan inspectie). Of wanneer de systeemarchitectuur (voorlopig) is gedefinieerd, moet men de mogelijke risico's en hoe deze te beheersen overwegen, bijvoorbeeld door middel van een proof of concept. Een laatste voorbeeld van risicobeheersing is door specifieke soorten beheersing toe te kennen aan geïdentificeerde (product) risico's. Lees meer over risicobeheersing en de maatregelen die genomen kunnen worden als onderdeel van de teststrategie in Building block 7 'Teststrategie'.

Maar om te beginnen, hoe bepaal je het risiconiveau nou helemaal? Een goed startpunt is tmap.net. Daar zijn hulpmiddelen te vinden als TMap Products Risk Analysis, PRISMA®, PRIMA® en Risk Poker in scrum. Hoewel er veel bruikbare aanpakken zijn, is de basis vaak vergelijkbaar. Kijk maar naar de definitie van productrisico, dat zal verschillen in formulering, maar niet in de betekenis:

#### Definitie

Een productrisico is de kans dat het product faalt in verhouding tot de verwachte schade als het dat doet:

Productrisico = Faalkans \* Schade

Soms wordt 'Faalkans' aangeduid als 'waarschijnlijkheid van voorkomen' en 'Schade' als 'impact'. Het maakt echt niet uit omdat het resultaat uiteindelijk vergelijkbaar, zo niet hetzelfde zal zijn.

We kunnen drie soorten risico's onderscheiden. Wanneer het primaire effect van het potentiële probleem de productkwaliteit betreft, worden ze "kwaliteitsrisico" of "productrisico" genoemd. Wanneer het primaire effect van het potentiële probleem het proces van de organisatie betreft, wordt het aangeduid als een "procesrisico". Wanneer de primaire gevolgen van het potentiële probleem het succes van het project betreft, wordt het aangeduid als "projectrisico" of "planningsrisico".

Het bovenstaande betreft het identificeren van risico's en de risicobeheersing. Soms kijken mensen verschillend tegen dezelfde zaken aan. Ze proberen vragen te beantwoorden zoals: wat is het belang van een bepaald product? Of: wat is de urgentie van een bepaald product?

Dit is natuurlijk belangrijk. Het product levert de meeste toegevoegde waarde als het product wordt gerealiseerd op het juiste moment. En als dit niet goed gebeurt kan het een enorm negatieve impact hebben op het succes van het project (dat is een voorbeeld van een plannings- / projectrisico). Dit kan ook worden gecombineerd met de productrisicoanalyse. Voorbeelden van dit soort geïntegreerde benaderingen zijn 'Product Risico en Baten Analyse' (PRBA) of 'Risico en Waarde' analyse.

## 2.7 Building block 7: Teststrategie

De opdracht en de productrisicoanalyse samen met de teststrategie vormen de basis van vrijwel alle testactiviteiten (zie de Building blocks 3 'Opdracht' en 6 'Productrisicoanalyse'). De productrisicoanalyse bevat de legitimiteit met betrekking tot wat er moet worden getest en welke risico's inherent aan het proces zijn. De teststrategie bepaalt welke van deze zou moeten worden beheerst en hoe. Dit kan de prioriteiten in een project beïnvloeden in de zin van bijvoorbeeld hoe groter het risico, hoe hoger de prioriteit (en gewenste beheersing). Besluiten wat wel of niet moet worden uitgevoerd, met betrekking tot tijd, kosten en baten (ook wel resultaten of klantwaarde) kunnen ook worden beïnvloed. Een veel gebruikte definitie voor teststrategie is de volgende:

### Definitie Teststrategie

De verdeling van het testwerk en dekking van de testobjecten is gericht op het zo vroeg en zo goedkoop mogelijk vinden van de belangrijkste fouten.

Deze definitie is nauw verwant aan de beheersing van risico's. Maar hoe zit het met de andere projectaspecten zoals kosten, tijd en baten? Immers, elk project focust op één of meerdere van deze aspecten (zie Building block 3 'Opdracht'). Deze focus moet worden vertaald in concrete keuzes in de teststrategie. Een nadrukkelijke keuze voor één van deze aspecten heeft vaak een effect op de andere aspecten. Als er bijvoorbeeld een maximum budget geldt worden misschien niet alle geïdentificeerde risico's op de gewenste manier beheerst, wat ervoor kan zorgen dat de acceptanten bepaalde restrisico's moeten tolereren bij de release naar productie. Of als er bijvoorbeeld een systeem wordt opgeleverd met een torenhoge zakelijke waarde, kritiek voor de missie en de veiligheid, dienen de risico's grondig te worden beheerst. Dit kan echter extra tijd en kosten vragen. Naast dit alles moeten de volgende principes vaak ook worden toegepast:

- Fouten moeten zo snel mogelijk worden gevonden na het fountinjectiepunt (lagere herstelkosten, snelle leercurve).
- Hoe groter het risico, hoe intensiever de test.
- Geen risico, geen test.

Vergeet niet dat dit *principes* zijn. 'Geen risico, geen test', bijvoorbeeld, is iets dat niet zal optreden in de praktijk bij de bouw van een stuk software. Er zal altijd een zeker risico zijn, of het stukje software is het überhaupt niet waard om te ontwikkelen.

Al deze overwegingen kunnen worden vastgelegd in een strategietabel. Net zoals er vele aanpakken zijn voor het uitvoeren van een productrisicoanalyse, zijn er ook vele soorten teststrategietabellen. De volgende lijst van teststrategieaspecten is op geen enkele manier volledig of verplicht. Het levert alleen een start-up lijst van die aspecten, waarvan zaken kunnen worden verwijderd, gewijzigd of toegevoegd. Aspecten die in een teststrategietabel opgenomen zouden kunnen worden (zie figuur 4 'Teststrategietabel'):

- Risico  
Risico is een generieke term, maar kan bestaan uit: testrisico's, productrisico's, projectrisico's of planningsrisico's.
- Risiconiveau  
Het niveau van het risico zoals besloten door alle belanghebbenden tijdens een productrisicoanalyse.
- Wanneer  
Eén van de teststrategieprincipes is: "Hoe eerder een fout kan worden gevonden, hoe beter – indien het nuttig is". Het is dus belangrijk om te beslissen wanneer een kwaliteitsmaatregel moet worden uitgevoerd.

- Locatie / Door wie  
Dit toont duidelijk aan, waar en bij wie de verantwoordelijkheid ligt voor de uitvoering van de kwaliteitsmaatregel(en).
- testvariëteit  
Een testvariëteit vertegenwoordigt een bepaalde behoefte aan testen, ongeacht hoe deze wordt georganiseerd. Andere termen kunnen worden gebruikt, zoals testsoort, testvorm, etc.
- (Kwaliteits-)Maatregel  
Om een specifiek risico te beheersen, worden één of meer kwaliteitsmaatregelen toegekend. Tijdens het toewijzen van de kwaliteitsmaatregel(en), wordt expliciet rekening gehouden met de omvang van het risico.

Natuurlijk moet je de tabel volledig aanpassen aan je eigen situatie!

Risico (betreffende)	Risico niveau	Wanneer	Locatie / doorwie	Test variëteit	(kwaliteits-) maatregel
Architectuur	hoog	Ontwerp fase	Team	Nvt	Proof of concept
Test Kennis:	laag	Project start	Sogeti	Nvt	
User Story x	Hoog	Ontwerp fase	Team	Evaluatie	Inspectie
		Testfase	Team	Acceptatietest	DV: paden
Performance	Medium	Realisatiefase	Derde partij	Performance test	DV: load profiel
Wijzigingsverzoek y	laag	Ontwerp fase	Team	Evaluatie	Review
		Testfase	Team	Acceptatietest	CT: Data rij
Requirement z	Hoog	Ontwerp fase	Team	Evaluatie	Inspectie
		Testfase	Team	Acceptatietest	nvt

Figuur 4. Teststrategietabel (DV: Dekkingsvorm)

## 2.8 Building block 8: Performance testen

Mensen maken vaak gebruik van de stelling “geen risico, geen test”. Voor de performance van IT-systeem bestaat de “geen risico” situatie eigenlijk gewoon niet meer. Optimalisatie van performance is kritiek wanneer de gebruikerservaring, zelfs in het meest triviale systeem, kan worden beïnvloed door slechte performance. Deze optimalisatie wordt nog belangrijker met de integratie van technologie in alle aspecten van ons professionele en persoonlijke leven (dat wil zeggen, cloud-based computing, mobiele oplossingen en het internet of things).

Bij performance testen is het organisatorische aspect wellicht het belangrijkste. De IT-organisatie (bijvoorbeeld in de vorm van een Performance Test Expertise Centrum) moet in staat zijn om organisatiebrede veranderingen te ondersteunen in ontwerp-, ontwikkel-, test- en onderhoudspraktijken met betrekking tot de performance van het systeem. Dit vereist het vermogen om instrumenten en methodieken te implementeren die voldoen aan project-specifieke behoeften en eisen (Agile, niet-Agile en onderhoud). Vooral bij Agile moet performance testen werkprocessen kunnen ondersteunen in sprints, of over sprints of teams heen. Deze performance testinspanning kan worden ondersteund via een gestructureerde performance testaanpak, waardoor u gemakkelijk toegang krijgt tot hulpmiddelen en resourcing van een resource pool van specialisten.

Performance testen is mogelijk in verschillende testvariëteiten, met een groot verschil in de aard van de gebruikte instrumenten, benodigde vaardigheden en, belangrijker nog, de diepte van het testen en het analyseren. Hieronder worden de belangrijkste Performance testvariëteiten weergegeven met sommige (maar niet alle) van de vereiste activiteiten:

### **Ontwerp voor Performance testen**

Van oudsher was het de praktijk bij performance testen om requirements te verzamelen en die met de performance aspecten uit te breiden. Dit leidde bij veel performance testactiviteiten tot weten wanneer de performance prestatie (zeer) slecht zou zijn. Ontwerpen voor performance legt de focus op het ontwerpen voor een goede performance, en het verstrekken van SMART requirements als input voor het valideren daarvan. De Performance testinspanning is niet actief betrokken bij alle aspecten van Design for Performance, maar vormt het kader voor het vastleggen van performancegerelateerde requirements in een vroeg stadium.

Activiteiten:

- Zorg ervoor beschrijvingen van klikpaden of de workflow op te nemen in de UI en het applicatieontwerp: Dit geeft nuttige en SMART requirements voor performance testen in alle stadia van de applicatie levenscyclus.
- Aandacht voor de performance-aspecten: van een Productrisicoanalyse gericht op specifieke aspecten van het systeem (in plaats van alleen 'Performance heeft een hoge prioriteit') naar betrokkenheid van eindgebruikers in het ontwerp van klik-paden en de verwachte intensiteit van gebruik van documenten (dit is ook een belangrijke verantwoordelijkheid voor de Product Owner).
- Ontwerpen voor performance testbaarheid: zonder afbreuk te doen aan de kwaliteit, kunnen ontwerpkeuzes in alles van de database tot de veiligheidsaspecten de optimale benutting van de beschikbare performance test tooling faciliteren (of een signaal geven dat er behoefte is aan extra tooling).

### **Ontwikkelen voor Performance testen**

Een project kijkt vanuit het meest technische perspectief naar performance testen op een componentniveau (unittesten), evenals op systeenniveau (integratietesten). Dit resulteert in het werken met platformspecifieke best practices voor ontwerp en ontwikkeling. Met de verschillende ontwerpkeuzes (en de gevolgen voor de performance) voor alles van ERP-systeem tot portal of mobiele oplossingen, is er geen eenduidige aanpak te noemen. Voortdurende waakzaamheid is nodig om gelijke tred te houden met nieuwe versies van de ontwikkelingskaders en de daaruit voortvloeiende performance effecten (zowel goed als slecht). Deze aanpak van 'Ontwikkelen voor Performance (Testing)' wordt vervolgens tijdens de Unit / Integratie testfase voor de eerste keer gevalideerd door specifieke componenten (web services / toegang tot de database etc.) te testen.

Activiteiten:

- Ontwikkelen voor de performance: het gebruik van best practices (uit de markt, maar ook bedrijfsspecifiek) met een focus op duidelijk gedefinieerde applicatiecomponenten (i.e. Servicelaag ontwerppatronen).
- Platformspecifieke (inclusief database) tooling en opleiding: Testgedreven ontwikkeling. Indien van toepassing, moet een performance test worden ontworpen en uitgevoerd als onderdeel van de unit- en integratietest.
- Implementatie van specifiek ontworpen stubs of simulatiesoftware voor nog niet beschikbare systeemcomponenten.

## **Acceptatie Performance testen**

Dit wordt beschouwd als de traditionele benadering van performance testen. Men maakt Load modellen en Iteratiemodellen op basis van gebruikersprofielen en klikpaden door de applicatie. Deze modellen moeten matchen met de verwachte belasting door de gezamenlijke gebruikspatronen van een grote groep eindgebruikers.

Activiteiten:

- Implementeren en gebruiken van Performance Test Tooling.
- Netwerkverkeer capture en playback tooling: Het ontwerpen, bouwen en onderhouden van de (productie-identieke) testomgeving en het gebruik van de beschikbare instrumenten om individuele performance testscripts te creëren.
- Multi-load generator controllers: het ontwerpen en implementeren van het performance-scenario (mix van testscripts) om de systeembelasting te simuleren zoals in de requirements gedefinieerd.

## **End-to-End Performance testen**

Voortdurende aandacht voor optimalisatie van performance vereist het testen van de verschillende applicaties op het totale landschap. Een permanente testorganisatie (bijv. PerformanceTest Expertise Centrum) kan voor meerdere applicaties scenario's combineren en scripts hergebruiken in een end-to-end Performance Test.

Activiteiten:

- Implementeren en gebruiken van de Performance Test Tooling zoals gedefinieerd in Acceptatie Performance Testen.
- Testlaboratorium opzet: ontwerp en onderhoud een laboratorium met voldoende capaciteit of flexibiliteit om meerdere applicaties te testen via een testscenario (extra vaardigheden nodig voor netwerkcomponenten, virtuele machine management, netwerkervaring, enz..).
- Omgevingsmonitoring: laten lopen en onderhouden van vergelijkbare tooling zoals die gebruikt worden in de productieomgeving, met de mogelijkheid om de Performance Test Tooling aan die resultaten te verbinden.

## **Productie Performance monitoring**

Performance bewaking in de productieomgeving dient een aantal verschillende doelen. Het primaire doel is om de bedrijfsprocessen te waarborgen en te zorgen voor een vroegtijdige waarschuwing bij performance degradatie. Dit wordt gedaan door het bewaken van de beschikbare resources van de gehele IT-infrastructuur. Door het uitvoeren van performance testscripts (voor een zeer beperkt aantal gesimuleerde gebruikers) in de productieomgeving, kan de performance ervaring van de eindgebruiker ook worden bewaakt.

Een secundair doel voor het uitvoeren van testscripts en resultaatbewaking in de productieomgeving is om feedback te geven aan eerdere performance testsoorten. Zijn de ontworpen testen nog steeds een accurate weergave van het gedrag van gebruikers? Dit voorkomt een situatie waarin de test- en controle opzet niet meer op het daadwerkelijke gebruik lijkt. Wanneer meer en meer verkeer wordt gegenereerd uit mobiele apparaten, moet de resulterende belasting parallel lopen aan het traditionele (PC-gebaseerde) browsergebruik in alle variëteiten van performance testen.

Activiteiten:

- Multi-load generator controllers: test scenario's uitvoeren met een lage impact (misschien buiten de piekuren) die continu de performance resultaten weergeven.
- Omgevingsmonitoring: laten lopen en onderhouden van monitoring tools en het verstrekken van rapporten / resultaten die kunnen worden vergeleken met de huidige testresultaten in eerdere stadia van de applicatielevenscyclus.

## **2.9 Building block 9: Test approaches**

Nadat men keuzes heeft gemaakt over wat er moet worden getest en hoe grondig specifieke onderdelen moeten worden getest, is de volgende stap: het bepalen van hoe daadwerkelijk de tests uit te voeren. Er zijn twee benaderingen voor het uitvoeren van testen: ervaringsgebaseerd en dekkingsgebaseerd.

### **Ervaringsgebaseerd**

Ervaringsgebaseerd testen laat de tester vrij om testgevallen vooraf te ontwerpen of om ze ter plekke te maken tijdens de testuitvoering. Deze tests zijn gebaseerd op de vaardigheden, intuïtie en ervaring van de tester. Onder Ervaringsgericht testen kan vallen:

- Checklist-gebaseerd: de ervaren tester maakt gebruik van een high-level lijst van items die moeten worden opgemerkt, gecontroleerd of herinnerd, of een set van regels of criteria waarop een product moet worden geverifieerd.
- Error guessing: op basis van de ervaring van de tester, gaat deze op zoek naar foutgevoelige plekken in het systeem en bedenkt daar geschikte testcases voor.
- Exploratory: het gelijktijdig leren, ontwerpen en uitvoeren van de tests; met andere woorden: elke testvorm waarbij de tester zijn testgevallen ontwerpt tijdens de testuitvoering en de verkregen informatie wordt hergebruikt om nieuwe en verbeterde testgevallen te ontwerpen. Exploratory testen kan zeer goed worden aangevuld met testen gebaseerd op dekkingsvormen.

### **Dekkingsgebaseerd**

Dekkingsgebaseerd testen is een manier om testsituaties af te leiden en te selecteren gebaseerd op een analyse van de testbasis en het toepassen van geselecteerde dekkingsvormen om een gewenste dekking te bereiken. Dekking heeft alles te maken met de wens om informatie over de kwaliteit en risico's efficiënt en effectief te verzamelen, en om de grootst mogelijke aantal afwijkingen, met zo min mogelijk testcases, gericht op specifieke aspecten van het testobject te vinden. Een dekkingsvorm is gericht op het bereiken van een specifieke dekking om specifieke foutsoorten op te sporen.

Er zijn grofweg vier groepen van soorten dekking (dekkingsvormen).

1. Proces: processen kunnen worden geïdentificeerd op verschillende niveaus. Er zijn algoritmen van controle flows en bedrijfsprocessen. Dekkingsvormen zoals paden, statement coverage, en state transitions kunnen gebruikt worden om (variaties in) deze processen te testen.
2. Condities : in bijna elk systeem zijn er beslispunten waar het gedrag van het systeem verschillende richtingen op kan gaan, afhankelijk van de uitkomst van een beslispunt. Afwijkingen van deze condities en de resultaten daarvan kunnen worden getest met behulp van dekkingsvormen als Decision coverage, Modified Condition/Decision coverage en Multiple Condition coverage.
3. Gegevens: gegevens worden gecreëerd en eindigen wanneer ze worden verwijderd. Tussendoor worden de gegevens gebruikt door ze bij te werken of hen te raadplegen.

Deze gegevenslevenscyclus kan worden getest, evenals combinaties van invoergegevens en de rubrieken van gegevensinvoer of -uitvoer. Grenswaarden, CRUD, Informatiestromen en Syntax zijn voorbeelden van soorten dekking in deze context.

4. Voorkomen: de manier waarop een systeem functioneert, hoe het presteert, wat de presentie moet zijn, wordt vaak beschreven in termen van niet-functionele requirements. Binnen deze groep vinden we soorten dekking zoals operationele en load-profielen en presentatie.

## **2.10 Building block 10: Crowd testen**

Bij crowd testen, of crowdsourced testen, is een virtuele groep testers over de hele wereld (een crowd) betrokken bij het testen, in plaats van een traditioneel beheerd testteam op één locatie.

'Crowd testen' heeft zijn wortels in 'crowd sourcing'. De sleutel tot het succes van crowd sourcing is dat er veel meer ideeën kunnen worden gegenereerd binnen een grotere groep mensen, en die ideeën elkaar kunnen beïnvloeden en elkaar steunen. Huidige Cloud en Web 2.0 technologieën maken het delen van ideeën in grote groepen mogelijk. Hetzelfde voordeel geldt voor crowd sourced testen: veel testers kunnen op tal van plaatsen bugs zoeken, en de bugs die een tester vindt kan andere testers beïnvloeden om te zoeken naar soortgelijke bugs.

Crowd testen heeft een tweede voordeel. Niet alleen de ideeën en ervaringen van de verschillende leden zijn meer divers, maar ook de verschillende hard- en software configuraties waartegen de onderzochte software wordt getest. Crowd testen maakt het mogelijk software te testen op een groot aantal apparaten, browsers en besturingssystemen, waarbij alle tests in parallel worden uitgevoerd, en daarmee de doorlooptijd van de tests geminimaliseerd worden.

Het managen van een grote groep van testers brengt andere uitdagingen mee dan het managen van een traditioneel testteam. Overwegingen om rekening mee te houden:

### **Testers of eindgebruikers**

Wilt u dat uw crowd actief op zoek gaat naar bugs of wil je dat ze de software gebruiken zoals ze dat in het echte leven zouden doen? In het eerste geval, neem ervaren testprofessionals als crowd testers. In het tweede geval, neem typische eindgebruikers.

### **Organiseer je eigen test crowd of gebruik een crowd testbedrijf**

In sommige gevallen is het niet mogelijk de software buiten de grenzen van uw eigen bedrijf vrij te geven. Financiële instellingen bijvoorbeeld hebben een probleem met het vrijgeven van software aan een grote menigte van relatief onbekende mensen. In een dergelijke situatie kan men er voor kiezen een crowd test te organiseren met de eigen medewerkers (een gesloten beta test). Indien er geen belemmeringen zijn om een testversie van de software buiten het bedrijf vrij te geven, zijn er talrijke bedrijven beschikbaar met een wereldwijd publiek van testers, waarbij het aantal testers kan oplopen tot honderdduizenden.

### **Belonen van de crowd**

Er zijn verschillende manieren om de crowd te managen. Iets om te overwegen is: hoe kan ik mijn testers belonen?

Bij een gesloten beta test binnen een bedrijf, wordt er vaak beloond voor het uitvoeren van de hele test. Bijvoorbeeld, als uw personeel een nieuw apparaat test, mag men het apparaat na de test houden.

In het geval van een publieke crowd kunnen er andere manieren worden ingezet om testers te belonen. Twee mogelijkheden die vaak worden gebruikt:

- Betaling voor elke gevonden bug.
- Betaling voor elke uitgevoerde test case.

Een nadeel van het belonen van de crowd voor elke bug is dat met een primaire focus op het vinden van bugs het moeilijk kan zijn om een goed beeld van de algehele kwaliteit van het te testen systeem te krijgen.

### **Nadelen van crowd testen**

Crowd testen biedt een aantal duidelijke voordelen zoals hierboven vermeld. Zoals crowd testen zich momenteel ontwikkelt, kleven er echter enkele nadelen aan die het gebruik ervan beperkt:

- Vertrouwelijkheid: hoe groter de crowd is en hoe meer die buiten de invloedssfeer ligt, hoe moeilijker het wordt om geheimhouding te managen.
- Kennis: niet iedere toepassing is geschikt voor crowd testen omdat er specifieke product- en proceskennis van de onderneming vereist kan zijn om de software uit te kunnen voeren.
- Testers die betaald worden 'per bug' gaan vaak op zoek naar de makkelijkst te vinden bugs in plaats van op zoek te gaan naar de meest kritische.
- Het bereiken van totale testdekking kan moeilijk zijn met crowd-testen.

## **2.11 Building block 11: Testvariëteiten**

Bij het organiseren van testen, hielden de traditioneel werkende testmanagers zich aan het hiërarchisch structureren van de testactiviteiten, op basis van de kwaliteitsattributen (kenmerken). Maar een testmanager onderscheidde ook vaak verschillende stadia. Gedefinieerde termen zoals Testsoort, Testvorm, Testfase en Teststadium werden vaak gebruikt.

In de hedendaagse visie op het testen, zijn bij testen betrokken mensen huiverig om het woord 'testsoort' te gebruiken, omdat het lijkt te impliceren dat verschillende groepen, gebaseerd op verschillende hiërarchische verantwoordelijkheden, verschillende tests zullen uitvoeren zonder enige interactie tussen deze testsoorten.

Bovendien hebben veel testers vaak moeite om onderscheid te maken tussen testsoorten en testvormen. En een testfase - is dat identiek aan een testsoort of niet?

Wat moet onze focus zijn bij het organiseren van testen?

Alle testactiviteiten moeten gezamenlijk alle belangrijke gebieden en aspecten van het systeem afdekken: dat is de belangrijkste doelstelling.

Om verwarring te vermijden over hoe we testtaken onderscheiden, introduceren we de term Testvariëteit.

#### Definitie

Testvariëteit is gericht op het bewust maken van alle stakeholders dat er altijd verschillende behoeften voor testen zijn, en er dus verschillende variëteiten testen zullen moeten worden georganiseerd. Of deze afzonderlijk of gecombineerd georganiseerd worden is afhankelijk van de situatie.

Er kunnen vele redenen zijn om diverse testvariëteiten te hebben. Zo zijn er verschillende stakeholders die moeten worden betrokken: programmeurs hebben een andere testfocus dan business vertegenwoordigers. Dit is vaak gerelateerd aan verantwoordelijkheden en de verantwoording over de testactiviteiten. De kwaliteitsattributen die moeten worden getest vormen een andere reden om onderscheid te maken tussen verschillende testvariëteiten. 'Onderhoudbaarheid' bijvoorbeeld vergt heel andere testactiviteiten dan 'Bruikbaarheid'.

Traditioneel werden verschillende kwaliteitsattributen afzonderlijk benaderd als een groep van testactiviteiten die samen werden ondergebracht in een testsoort. Veel mensen kennen de 'functionele acceptatietest', waarvan de naam al aangeeft dat het testen incompleet was, omdat het blijkbaar niet gericht was op niet-functionele kenmerken. In de nieuwe zienswijze kunnen functionele en niet-functionele tests worden beschouwd als testvariëteiten. Afhankelijk van de omstandigheden, zoals het application lifecycle model dat wordt gehanteerd, worden deze testvariëteiten samen of afzonderlijk georganiseerd. Het belangrijkste is dat alle relevante testvariëteiten op de een of andere manier worden uitgevoerd.

Onervaren Agile teams hebben de neiging om hun testinspanningen op unit- en systeem testen te richten; dat wil zeggen testen of computerprogramma's voldoen aan de technische eisen. Dit is absoluut een van de belangrijke testvariëteiten. Maar een ander belangrijk aspect is om te valideren of de zakelijke doelen zijn gehaald, de 'acceptatie test', die kan worden gedaan door de product owner in een Agile team. Niet alle Agile teams besefven het belang van deze testvariëteit. In de praktijk moet deze testvariëteit door het Agile-team in dezelfde iteratie worden uitgevoerd, zodat de testvariëteiten in dit voorbeeld samen georganiseerd worden, ook al kunnen verschillende leden van het team aan verschillende testvariëteiten werken. Natuurlijk zal het Agile team ook diverse testvariëteiten onderscheiden zoals performance testen en security testen, die ook tijdens de iteratie uitgevoerd kunnen worden.

Echter, vooral in een grotere organisatie, zal er vanuit de stakeholders behoefte zijn aan testvariëteiten die niet door één team kunnen worden uitgevoerd, maar apart georganiseerd dienen te worden, zoals een end-to-end test.

Als je ooit hebt deelgenomen aan een discussie over de vraag of een end-to-end test een testsoort is of een testvorm, zul je erkennen dat dit eigenlijk niet uitmaakt, zolang alle testactiviteiten met betrekking tot het end-to-end bedrijfsproces maar worden uitgevoerd.

Dus laten we de term testvariëteit gebruiken, om alle betrokkenen bewust te maken van het feit dat er verschillende standpunten zijn ten opzichte van testactiviteiten, en we ervoor kunnen zorgen dat de belangen van alle stakeholders worden afgedekt door deze op een weloverwogen manier aan te pakken.

## 2.12 Building block 12: Testmanager in Agile omgevingen

Testmanagers hebben de neiging nogal zenuwachtig over Agile te zijn. Wanneer de focus van een testteam verschuift naar samenwerking over producten en projecten heen, in

plaats van het testen van een geïsoleerd fase of dienst, kan het lijken alsof de behoefte aan een testmanager verdwijnt. Omdat testers hun vorderingen rechtstreeks in hun projectteams moeten communiceren, hun schattingen verstrekken als onderdeel van een Agile methodiek met behulp van een just-in-time testplanning, dan lijkt er geen behoefte aan een testmanager te zijn die optreedt als intermediair of opzichter op projectniveau. Maar hoe zit het met de andere testmanagement activiteiten? Hoe gaat Agile daar mee om?

#### Voorbeeld

Er zijn drie rollen in een scrum team: de product eigenaar, de scrummaster en de teamleden (of ontwikkelaars). Het team is zelforganiserend en multidisciplinair, zonder managers. Er is geen ruimte voor testmanagers in een dergelijk team. Testen is een rol die elk teamlid moet kunnen uitvoeren. Het feit dat 'testmanager' geen rol is in een scrum team betekent echter niet dat de testmanagement activiteiten niet moeten worden uitgevoerd. Integendeel, deze blijven even belangrijk. Maar ze kunnen worden uitgevoerd door een willekeurig lid van het team met de juiste kennis en vaardigheden. Toch is het raadzaam om een professionele tester in het team op te nemen, om de aanwezigheid van testdeskundigheid te garanderen. Deze tester (misschien een vroegere testmanager) heeft kennis van de uitvoering van een risico-analyse, het uitvoeren van evaluaties, testontwerptechnieken, het opstellen en uitvoeren van testcases, testautomatisering, etc. Het betekent echter niet dat alle testactiviteiten moeten worden uitgevoerd door deze tester. Andere teamleden kan bijvoorbeeld worden verzocht om steun bij de totstandkoming en uitvoering van de test cases te bieden. In een dergelijke situatie kan de professionele tester als coach optreden. Als een team geen voldoende testdeskundigheid kan garanderen, kan het een optie zijn om een testmanager toe te staan van buitenaf het team te ondersteunen en een coach voor de tester (s) te zijn.

Maar over het algemeen kan men in Agile omgevingen de testmanagerrol zien evolueren naar een hoger niveau waar de activiteiten onder andere zijn:

- In sprint nul: adviseur van het team - hoe om te gaan met de kwaliteitsverantwoordelijkheid.
- Faciliteren van de communicatie over de teams heen voor een groot aantal Agile-projecten binnen een organisatie.
- Presenteren van een geaggregeerde visie op testgebied voor hoger management.
- Persoonlijke ondersteuning, begeleiding en professionele ontwikkeling voor testers (bijvoorbeeld als lijnmanager).
- Fungeren als escalatiepunt voor testers.
- Budgettering en forecasting voor testing as a service (afhankelijk van het soort organisatorische proces – moet testing as a service worden gebruikt).
- Betrokken zijn bij scrum-of-scrum meetings.
- Het verstrekken van advies met betrekking tot de kwaliteit.
- Functioneren als stakeholder voor de product owner.
- Combinatie met de scrummaster rol.

### **2.13 Building block 13: Permanente testorganisatie**

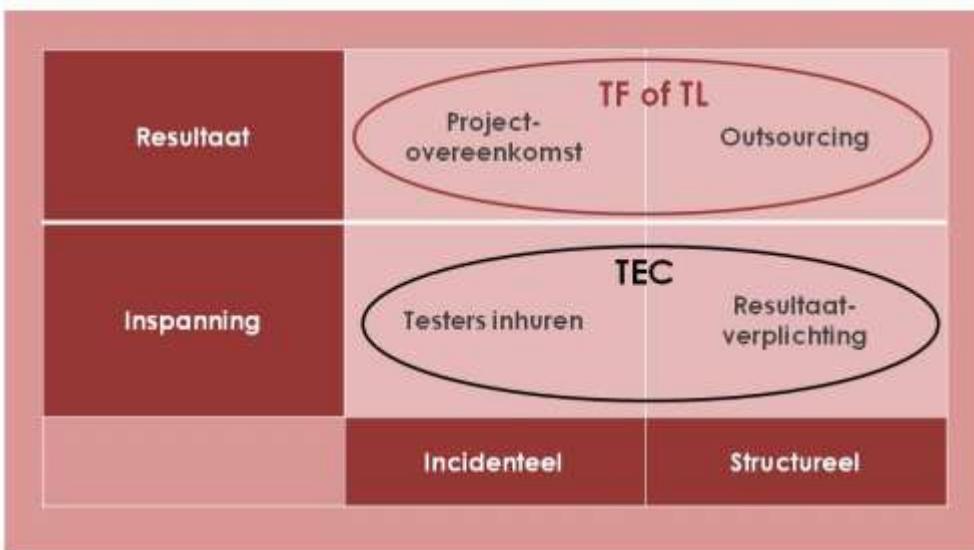
Twee vormen van permanente testorganisatie komen vaak voor in de praktijk. Deze zijn (zie figuur 5):

- De permanente testorganisatie als testexpertisecentrum (TEC).
- De permanente testorganisatie als testfabriek (TF) of testlijn (TL).

De twee zijn verschillend in onder andere de diensten die zij aanbieden en hun verantwoordelijkheid. De TEC (vaak geïmplementeerd als een "staforganisatie") is vooral een leverende en adviserende organisatie die een 'inspanningsverplichting' levert bij het aanbieden van diensten. Zo kan het testers of testmanagers leveren voor een project of zelfs voor een andere lijnorganisatie binnen het bedrijf. Of het kan adviezen geven over te gebruiken testaanpak of testtool (bijvoorbeeld voor een scrum of DevOps team). De activiteiten worden altijd uitgevoerd onder de verantwoordelijkheid van het project. De TF of TL geeft een 'resultaatverplichting' voor veel van haar diensten. Het proces kan worden vergeleken met een fabriek met vast personeel (testers), machines (infrastructuur), gestandaardiseerde werkprocedures, enzovoort. Verschillende klanten (afdelingen, projecten, systemen) kunnen hun volledige testopdrachten aan dit type testorganisatie uitsteden als ware het een 'lijnorganisatie'.

De term "Test Competence Center of excellence" duikt ook vaak op, wanneer we spreken over een permanente testorganisatie. Dit kan een van de genoemde organisatievormen zijn.

Beide testorganisatievormen baseren hun benadering op basis van de frequentie van de vraag in de testdiensten. Voor incidentele verzoeken ('het opzetten van een testomgeving') wordt een andere benadering gekozen dan voor structurele aanvragen ('testreleases').



Figuur 5. Twee veelvoorkomende vormen van permanente testorganisaties

## 2.14 Building block 14: Model based testen

Het creëren van testcases is de kern van testen in elke ontwikkelmethode en kan zeer tijdrovend zijn, zeker als het handmatig uitgevoerd wordt. Ook kunnen er bij testcaseontwikkeling interpretatiefouten worden gemaakt als de testbasis - of het nu de requirements, de ontwerpdокументen of enig ander artefact betreft – dubbelzinnig is.

Model Based Testen (MBT) kan onder andere bestaan uit volledige testautomatisering. Daarbij worden testgevallen vanuit modellen gemaakt en in één keer uitgevoerd door een MBT suite. Een andere mogelijkheid is Model Based Test design (MBTd). Deze aanpak is gericht op het verkorten van de ontwikkeltijd voor testcases. Een derde mogelijkheid is

Model Based Review (MBR). Dit is gericht op het verminderen van ambiguïteit in de testbasis, maar zonder het leveren van daadwerkelijke testgevallen.

### Model Based Review

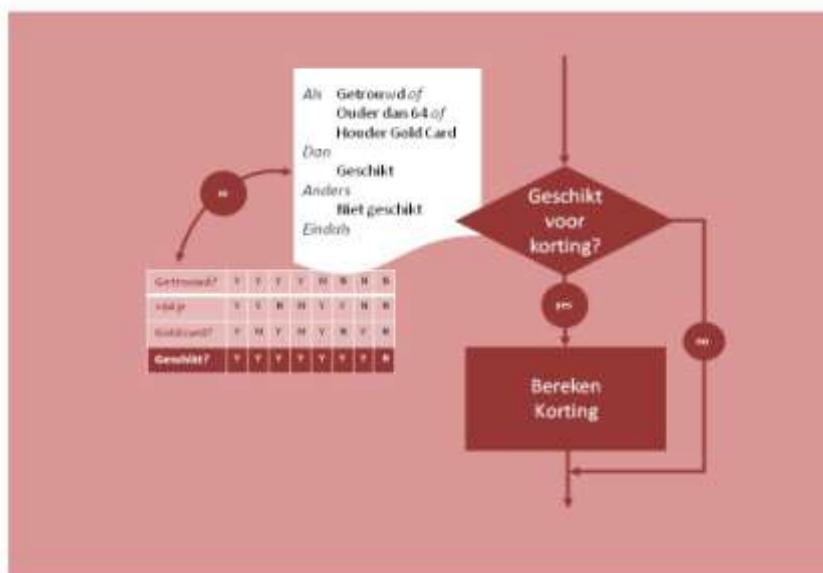
Modellen in MBR vormen een middel tot een doel, waarbij de duidelijkheid en compleetheid van de testbasis het doel is. De tester maakt één of meer modellen, zodat eindgebruikers, analisten, ontwerpers etcetera kunnen verifiëren in hoeverre de tester het onderwerp begrepen heeft. De bron kan bestaan uit tastbare documenten, maar ook 'in de hoofden' zitten van iedereen.

De kerngedachte achter MBR is dat modellen eenduidig zijn van aard, zodat gebreken zoals onvolledigheid, inconsistentie en onjuistheid gemakkelijker opvallen.

Modellen geven ook een beperkt zicht op de werkelijkheid, dus moeten er vaak meerdere modellen worden samengesteld om een compleet beeld te krijgen van wat er in de ontwerpartefacten of 'in de hoofden van' betrokkenen zit. Bijvoorbeeld, een werkwijze wordt het best weergegeven door een stroomdiagram, maar een "ja / nee" beslissing in dat proces kan bestaan uit diverse basale "ja / nee" voorwaarden. Deze voorwaarden kunnen individueel en expliciet worden gemodelleerd in het stroomschema, maar de voorkeur gaat hierbij uit naar een beslistabel of pseudo-code.

### Model Based Test design

MBTd (zie figuur 6) bouwt voort op de eenduidigheid van modellen: ze kunnen automatisch worden geïnterpreteerd en omgezet in testgevallen. De volledigheid en de mate van detail van het model bepaalt de mogelijkheid om fysieke testgevallen voor geautomatiseerde testuitvoering te genereren, logische testgevallen voor handmatig testen te maken, of iets daar tussenin. De architectuur van het te testen systeem is ook een belangrijke factor in de haalbaarheid van het eindresultaat: systeemtesten van een SOA-gebaseerde applicatie is een betere kandidaat voor automatisch uitgevoerde fysieke testgevallen dan End-to-End tests die vele systemen omvatten buiten de controle van de tester.



Figuur 6. Model Based Test design

De basis voor MBTd kan zowel bestaan uit design modellen of 'testmodellen' van MBR: (her) gebruik van design modellen is vaak de snelste en gemakkelijkste manier om MBTd te implementeren. Er is echter één aspect dat speciale aandacht verdient: bevatten de design modellen voldoende detail om de testdoelen van de tester te bevredigen? Dit is niet altijd het geval: indien bijvoorbeeld het testdoel is om naleving van design standaarden te verifiëren, is het zeer onwaarschijnlijk dat de design modellen expliciet deze standaarden modelleren! Aanvulling op de bestaande modellen met testen van specifieke details kan arbeidsintensiever blijken te zijn dan het maken van testgevallen zonder model.

### **Volledig automatisch Model Based Testen**

In zijn optimale vorm vermindert MBT de testinspanning voor het maken en reviewen van modellen, waarna één druk op de knop volstaat om tests te creëren en uit te voeren. Er zijn verschillende suites waarmee dit kan, maar een benadering met meerdere stappen kan ook haalbaar zijn, waarbij voor de verschillende stappen verschillende tools worden gehanteerd, en zo een gefaseerde implementatie van MBT mogelijk wordt gemaakt. Een mogelijke reden om te kiezen voor een gefaseerde aanpak is hergebruik van de reeds bestaande tools en frameworks.

Een zeer mooi 'neveneffect' van MBR is de geleidelijke overgang van pure testmodellen naar 'all-purpose-modellen', die kunnen worden gebruikt bij de analyse, het ontwerp en testen tegelijk, omdat ontwerpers na het reviewen van de modellen het eigenaarschap ervan overnemen. Dus MBT integreert in twee richtingen: testers gebruiken design modellen voor MBTd, en ontwerpers gebruiken de MBR testmodellen.

Misschien wel het grootste voordeel van MBT schuilt in beheer: het aanpassen van een (test of design) model en vervolgens genereren van tientallen of zelfs honderden testcases met een druk op de knop kan nooit worden geëvenaard door handmatig testcaseonderhoud: niet in doorlooptijd, niet in kosten en niet in kwaliteit!

MBR en MBTd hebben elk hun eigen individuele voordelen, maar de sterkste toepassing is de combinatie van de twee: op zijn best worden interpretatiefouten voorkomen en wordt handmatige testuitvoering vermeden.

## **2.15 Building block 15: Kwaliteitsbeleid**

In het algemeen hebben bedrijven die structureel gebruik maken van testen een testbeleid.

Bedrijven die kwaliteit belangrijk vinden hebben een kwaliteitsbeleid. 'Beleid' wordt hier gebruikt als overkoepelende term. Andere termen die in deze context gebruikt worden zijn 'missie', 'visie', 'strategie'.

Een kwaliteitsbeleid omvat keuzes gemaakt door het management die in het algemeen van toepassing zijn op de operationele bedrijfsvoering. Soms heeft het de vorm van een formeel en gecertificeerd kwaliteitssysteem. Ook dat is een managementkeuze. ISO9000 is een bekende standaard voor kwaliteitscertificering.

Een kwaliteitsbeleid zorgt ervoor dat een organisatie, product of dienst consistent is en niet alleen is gericht op de product- en dienstverleningskwaliteit, maar ook op het proces om dat te bereiken.

## Waaruit bestaat een kwaliteitsbeleid?

Een kwaliteitsbeleid is altijd gebaseerd op de strategie van de onderneming. Het bevat vaak onderwerpen als: visie op de kwaliteit, de doelstellingen, de reikwijdte en de kwaliteitsprincipes (bijvoorbeeld over klantgerichtheid, leiderschap, mensen, een systematische aanpak door middel van processen, gebruikte kwaliteitsnormen of modellen, continue verbetering, etcetera).

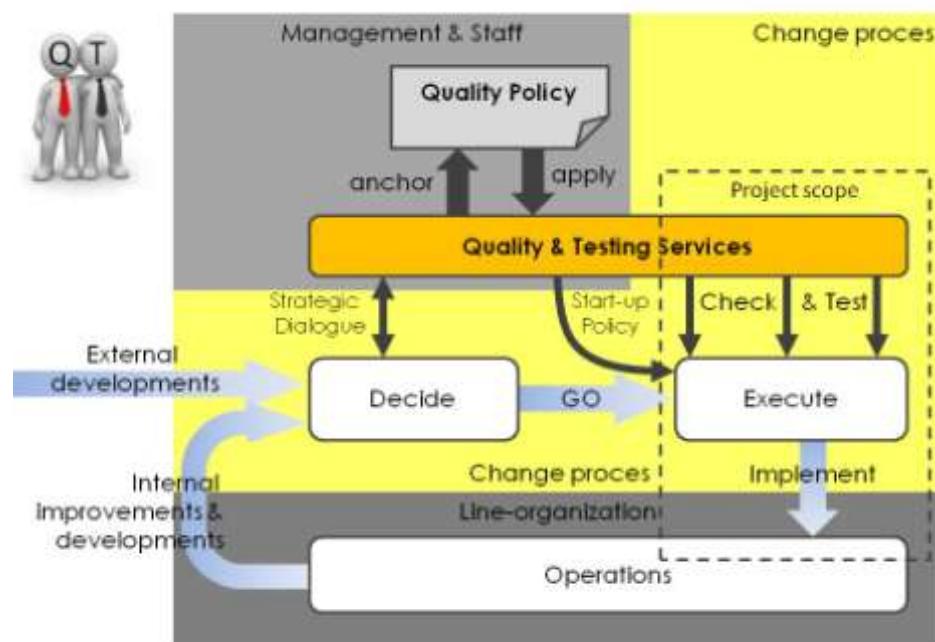
Als het gaat om IT-projecten zullen er thema's in voorkomen als: ambitieniveau op kwaliteitsgebied, continue verbetering, de gebruikte methoden, gemeenschappelijke tooling, hoe de kwaliteits- en testdeskundigheid wordt georganiseerd. In het algemeen: alle keuzes die projectoverkoepelend zijn.

## Wanneer heb je een kwaliteitsbeleid nodig?

Die beslissing is aan het topmanagement, maar over het algemeen is het nodig om ervoor te zorgen dat de kwaliteitsaspecten op dezelfde wijze worden behandeld in de gehele onderneming, op een manier die de waarden van het bedrijf weerspiegelt. Wanneer een beleid nodig is, dan wordt de redactie, toepassing en controle meestal toegewezen aan een kwaliteitsstafafdeling.

## Hoe maak en onderhoud je een beleid?

De volgende afbeelding (figuur 7) toont de betrokkenen bij het creëren, toepassen en handhaven van een kwaliteitsbeleid in een projectomgeving, gerelateerd aan het veranderingsproces.



Figuur 7. Kwaliteitsbeleidsproces in een projectomgeving

Het veranderingsproces staat centraal, in hoofdzaak bestaand uit twee stappen: een besluit nemen over hoe te reageren op de ontwikkelingen en de besluiten uitvoeren. De

figuur toont de kwaliteits en testdiensten die dit proces ondersteunen. Doorgaans toegewezen aan een QA stafafdeling, bestaan deze diensten bestaan hoofdzakelijk uit drie delen:

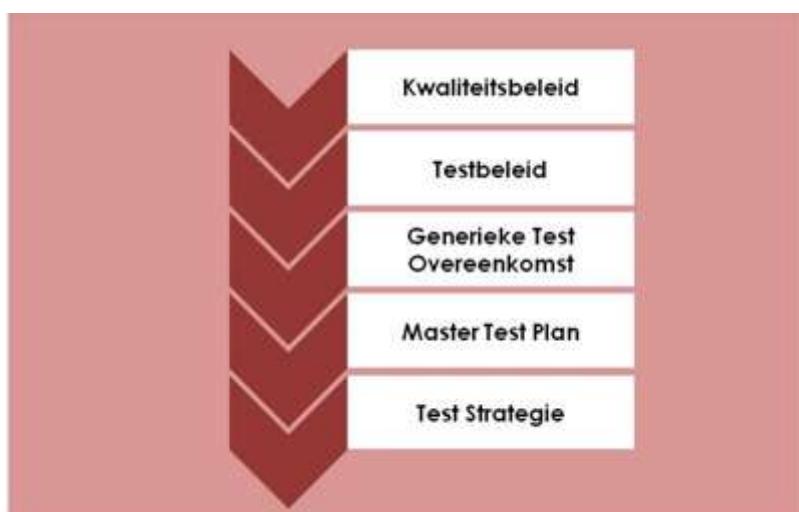
- Het beslissingsproces wordt ondersteund vanuit het kwaliteitsperspectief. Het bestaande beleid wordt toegepast, aangepast of uitgebreid in een strategische dialoog met het beslissende management.
- Wanneer de beslissing is genomen om een project te starten, wordt het project ondersteund door een start-up beleid.
- Tijdens de uitvoeringsfase wordt het onderhavige product of dienst in het project door middel van testen gecontroleerd op kwaliteit.

### **Hoe zorg je ervoor dat een IT-project het beleid daadwerkelijk toepast?**

Nadat de beslissing over de wijze waarop gereageerd wordt op de ontwikkelingen is genomen, worden de noodzakelijke veranderingen vastgesteld en worden de IT projecten geïnitieerd. Wanneer een project zich in de opstartfase bevindt, kan men uit het volledige beleid die zaken selecteren die van toepassing zijn op de doelstelling van het project: methoden, instrumenten, teststrategie, etcetera. In feite kan een kwaliteits- en testplan in combinatie met het projectplan worden opgesteld. Dit zorgt ervoor dat in de project scope, het budget en de planning rekening wordt gehouden met de benodigde kwaliteits- en testinspanningen.

### **Hoe verhoudt een kwaliteitsbeleid zich tot het testen?**

Een kwaliteitsbeleid bevat maatregelen om de kwaliteit van alles wat onderworpen is aan het beleid te controleren en aan te tonen. Testen is één van die maatregelen. Testen is een uitstekende maatregel om de werkelijke kwaliteit aan te tonen. In TMap bepaalt de teststrategie hoe het testen wordt aangepakt en dat is opgenomen in een master testplan. De zogenaamde Generieke Test Afspraken (GTA) lijken op een mastertestplan en vervangen dat soms zelfs. Een testbeleid vervangt GTA die kan worden afgestemd op specifieke uitbestedingssituaties. Een kwaliteitsbeleid geeft aan hoe testen wordt gebruikt als middel voor het meten en aantonen van kwaliteit (zie figuur 8).

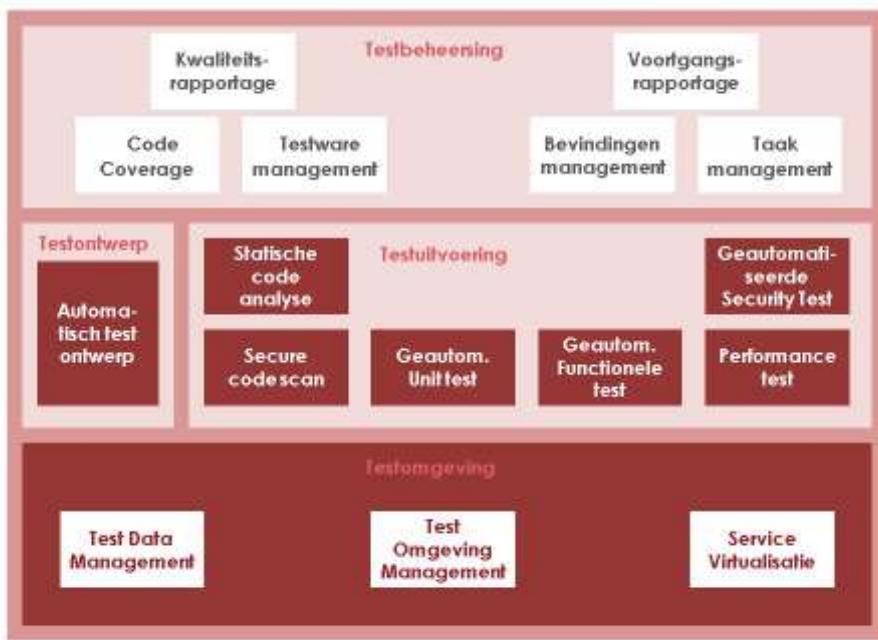


Figuur 8. Verhouding kwaliteitsbeleid tot testen

Succes van het kwaliteitsbeleid wordt met name bepaald door factoren als het verbeteren van commitment, kennis en expertise om verbetering te bereiken, de reikwijdte van de gewenste verbetering en de opname ervan in de bedrijfscultuur. Voor het opzetten van een kwaliteitsstrategie, is het belangrijk om het element van mensen toe te passen, met inbegrip van cultuur en teambuilding. Elke verbetering (verandering) kost tijd om te implementeren en als geaccepteerde praktijk te stabiliseren. Verbeteringen waarbij de bedrijfscultuur wordt aangepast duren langer, omdat ze meer weerstand moeten overwinnen. Het is gemakkelijker en vaak effectiever binnen de bestaande culturele grenzen te werken aan kleine verbeteringen (dat is Kaizen) dan grote veranderingen aan te brengen in een 'big bang' aanpak. Aan de andere kant, een 'big bang' verandering werkt het beste wanneer een onderneming wordt geconfronteerd met een crisis en grote wijzigingen moet aanbrengen om te kunnen overleven. Een goed gedefinieerd kwaliteitsbeleid moet met al deze factoren rekening houden.

## 2.16 Building block 16: Gebruik van testtools

Er zijn veel verschillende soorten testtools, elk met een eigen doel. We kunnen testtools classificeren door vermelding van de testactiviteiten die ze ondersteunen (zie figuur 9).



Figuur 9. Testtool classificatie

Dit betekent niet dat deze individuele activiteiten worden ondersteund door een enkele testtool. De meeste testmanagementtools beschikken over een combinatie van bijvoorbeeld testware management, bevindingenmanagement en rapportagefuncties.

Het gebruik van deze hulpmiddelen is gericht op een effect. Hierbij is het zinvol om onderscheid te maken naar primaire en afgeleide effecten (zie figuur 10). Uitvoeringstesttools versnellen testuitvoering, zodat het primaire effect tijdsbesparing is. Er bestaat een keuze in de te bereiken effecten: ofwel vermindering testuitvoeringsijd, of toenemende dekking in dezelfde testuitvoeringsijd, of verhoging van het aantal keren dat

de tests wordt uitgevoerd. De uitzondering in deze categorie van testtools is een performance testtool, waarvan het primaire effect is de mogelijkheid om een performancetest uit te voeren en het afgeleide effect is inzicht in performance en stabiliteit.

De andere soorten testtools hebben verschillende soorten effecten. Testcontroletools hebben het primaire effect van kwaliteits- en voortgangsbeheersing, testdesign tools besparen tijd en testomgeving tools leveren beheersing op over de randvoorwaarden om tests uit te voeren.

De primaire en afgeleide effecten van de meest gebruikte types testtools worden hieronder weergegeven. Bij meerdere afgeleide effecten dient een keuze te worden gemaakt welke effecten men nastreeft.

Type tool	Ondersteunde activiteit	Primaire effect	Uiteindelijk effect
Test management tool	<ul style="list-style-type: none"> <li>• Toetsware management</li> <li>• Bevindingenmanagement</li> <li>• Kwaliteitsrapportage</li> <li>• Voortgangsrapportage</li> </ul>	Beheersing over testproducten	Kwaliteitsbeheersing
Code coverage analyse tool	Code dekking	Inzicht in testdekking op codeniveau	Kwaliteitsbeheersing
Model based testing tool	Geautomatiseerd testontwerp	Tijdsbesparing	<ul style="list-style-type: none"> <li>• Verminderen testontwerptijd</li> <li>• Vergroten dekking van de testuitvoering</li> </ul>
Static code analyse tool	Statische code analyse	Inzicht in codekwaliteit	Kwaliteitsbeheersing Technisch
Unit testing framework	Geautomatiseerde unit test	Tijdsbesparing	<ul style="list-style-type: none"> <li>• Vergroten dekking</li> <li>• Vaker testen</li> </ul>
Test uitvoering tool	Geautomatiseerde functionele test	Tijdsbesparing	<ul style="list-style-type: none"> <li>• Verminderen testuitvoeringstijd</li> <li>• Vergroten dekking</li> <li>• Vaker testen</li> </ul>
Performance test tool	Performance test	Uitvoeren load- en stress test	Kwaliteitsbeheersing • Performance • Stabiliteit
Test data management tool	Test data management	De juiste testdata hebben binnen de grenzen van privacy wetgeving/kosten voor de testomgeving	<ul style="list-style-type: none"> <li>• Verminderen testuitvoeringstijd</li> <li>• Vergroten dekking van de testuitvoering</li> </ul>
Service virtualisatie tool	Service virtualisatie	Verminderen van de afhankelijkheid van de beschikbaarheid van de service	<ul style="list-style-type: none"> <li>• Eerder testen</li> <li>• Parallel testen</li> <li>• Altijd testen</li> </ul>

Figuur 10. Type tools, toepassing en effect op korte en lange termijn

Kostenreductie is altijd een afgeleid effect. Het opmerkelijke is dat de belangrijkste financiële voordelen van het gebruik van testtools niet het testproces zelf betreffen: het verminderen van testtijd helpt de onderneming door eerder klantwaarde toe te voegen, het verbeteren van de kwaliteit biedt voordelen voor de bedrijfsvoering door vermindering van het aantal incidenten, en het vroeger vinden van fouten helpt ontwikkeling door verlaging van de herstelkosten.

De effecten van afzonderlijke soorten testtools kunnen worden verhoogd door ze te combineren of te integreren. En nog meer voordeel kan worden behaald door ze te combineren of ze te integreren met andere tools die gebruikt worden in de applicatie

levenscyclus, zoals tools uit het ontwikkelingsproces voor requirementsmanagement, systeemontwerp, -ontwikkeling of -implementatie, en instrumenten die gebruikt worden in beheer voor change of issue management.

## **2.17 Building block 17: Kenmerken kwaliteitsgestuurde aanpak**

Zoals in Musing 2 beschreven staat leiden de vier basiselementen van TMap HD leiden tot Vertrouwen, het vijfde element. De aanpak die dit creëert is gericht op de kwaliteit van het product. Dat is de reden waarom deze aanpak 'kwaliteitsgestuurd' wordt genoemd. Het kan worden geïntegreerd in allerlei ontwikkelings- of projectmethoden, kaders of benaderingen. Beter nog: de aanpak zal alleen succesvol zijn als het geïntegreerd wordt, aangezien Integreren een belangrijk element is en het kan niet werken als een alleenstaand proces.

De methode waarmee wordt geïntegreerd kan ervoor zorgen dat de elementen uit de kwaliteitsgestuurde aanpak enigszins anders worden toegepast. Sommige kenmerken zijn direct gerelateerd aan één element, anderen volgen uit een combinatie van elementen. In dit building Block geven we een voorbeeld van een aantal van de kenmerken van de kwaliteitsgerichte aanpak zoals deze is geïmplementeerd in het TMap HD boek "Neil's quest for quality" – deze kenmerken zijn bedoeld ter inspiratie voor iedereen die graag kwaliteitsgericht werken wil invoeren bij zijn of haar organisatie.

Veel kenmerken zijn op de een of andere manier met elkaar verbonden.

De kenmerken met betrekking tot testen zijn apart genoemd.

De volgorde van de lijst betekent niet noodzakelijk een rangorde van het relatieve belang, noch is de lijst volledig.

Kenmerken van een kwaliteitsgestuurde aanpak:

- Alleen functionaliteit die voldoet aan het vooraf gedefinieerde kwaliteitsniveau wordt vrijgegeven.
- Directe betrokkenheid van gebruikers en hun management (business-driven).
- Test in alle fasen, begin zo vroeg mogelijk.
- Test waar mogelijk en zinvol geautomatiseerd om beter, meer en vaker te testen.
- Testen op het einde is alleen bedoeld om de waarde aan te tonen, een werkende oplossing te laten zien.
- De rol van de testprofessionals evolueert: geïntegreerd met andere disciplines om hen in alle fasen, stadia en activiteiten te helpen met behulp van hun testdeskundigheid.
- Test wordt niet zozeer gebruikt om fouten te vinden, maar om goede werking aan te tonen. Het doel van testen verschuift meer naar: bewijsvoering.
- Kwaliteit gaat iedereen aan.
- Kwaliteit is ingebouwd in het proces.
- Continue verbetering van de werkwijze is ingebouwd
- De betrokken personen hebben mandaat om te beslissen over hun eigen werkproces om de kwaliteit te verbeteren.
- Elke afwijking, fout, onvolmaaktheid is een trigger om te verbeteren.
- Open cultuur waar mensen elkaar kunnen vertrouwen.
- Mindset: een houding om alle bovenstaande aspecten na te leven en te eren
- Een kwaliteitscoördinator heeft mandaat om in te grijpen op de aspecten van kwaliteit en voortdurend aandacht te besteden aan kwaliteit, gebruik makend van tests om dit te monitoren en te controleren.
- Ondersteuning vanuit het hoogste managementniveau.

Als een voorbeeld van hoe het in de praktijk kan werken, wordt het korte actieplan van Seabiscuit getoond, gemaakt door Neil, Hal en Francine (de karakters in "Neil's quest for quality") en gebruikt in het verhaal:

	Instrumenten, maatregelen en acties om dit te bereiken	Actie door:
-	Bezoek aan een fabriek ter inspiratie.	H (M)
-	Overall projectmanagement.	H
-	Huur teamleiders in, die ervaren zijn in het op een kwaliteitsgestuurde wijze bouwen van een team en te coachen.	H
-	Agile methode (kortcyclisch, iteratief, met behulp van demo, retrospectief en Definition-of-Done).	H
-	Teamretrospectieven om het proces te verbeteren bij elke afwijking van de kwaliteitsstandaard.	H
-	Training en teambuilding.	H
-	Coaching van Hal door dhr. Mikkel gericht op de kwaliteitsgestuurde aanpak.	M
-	Selecteer mensen zorgvuldig om teams te bouwen.	H (M)
-	Het betrekken van Ann als representatie van de gebruikers, deelnemend op een dagelijkse basis.	N
-	Teamoverstijgende retrospectieven door Neil (eventueel bijwonen en stimuleren van teamretrospectieven).	N
-	Continue aandacht voor kwaliteitsaspecten, (opstarten, communiceren, stimuleren)	N
-	Controle op kwaliteitsaspecten (dashboard).	N
-	Adequate set van kwaliteitscriteria, gebruikt als standaard voor het gedefinieerde kwaliteitsniveau.	N
-	Gebruik tests om te monitoren en de werkelijke toestand van de kwaliteit te controleren.	N
-	Het gebruik van testtools om meer, beter en vaker te testen.	N
-	Oppakken en beheersen van projectoverstijgende problemen, verbeteringen, ervaringen (PDCA), en het opbouwen van een lange termijn beleid voor ZBO (testdeskundigheid, tools, kwaliteitsgestuurde aanpak, enz.)	N
-	Positie van Neil onafhankelijk van het project, optredend namens Owen, ondersteuning gevend aan Ann en Hal, overzicht houdend. Gemandateerd door Rupert om in te grijpen op de kwaliteit kwesties (toewijzing).	N

De verantwoordelijkheid voor elk punt wordt aangegeven door initialen: H = Hal/Projectmanager, N = Neil/Testmanager, M = Mr. Mikkel(Coach).

Kwaliteitsgericht werken levert de volgende garanties op:

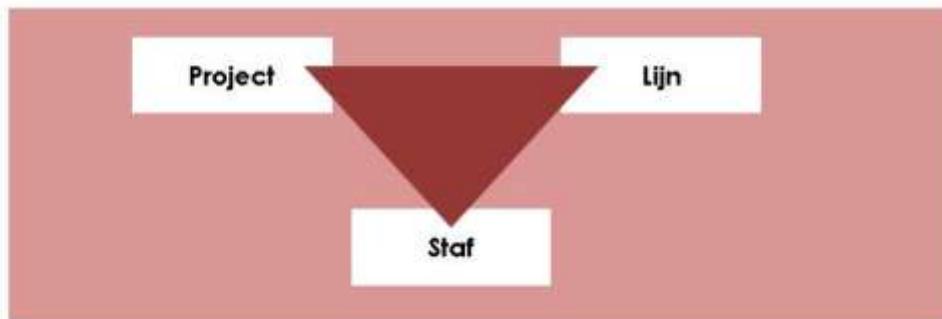
- Door alleen productiteraties vrij te geven die voldoen aan afgestemde kwaliteitscriteria, wordt hoge kwaliteit gegarandeerd.
- Er wordt aan het einde van elke timebox gegarandeerd een werkende oplossing opgeleverd.
- Door continue te verbeteren wordt gegarandeerd dat op termijn de kwaliteit gaat stijgen en de kosten en doorlooptijd gaan dalen.

- Als er na meerdere timeboxen een harde deadline is, wordt gegarandeerd dat op dat moment alle belangrijkste functies aanwezig zijn en werken. Dat komt omdat functies elke timebox opnieuw kunnen worden gewaardeerd.

## 2.18 Building block 18: Geïntegreerde testorganisatie

Hoe organiseer je het testen? Veel mensen hebben daar moeite mee. Veel verschillende oplossingen zijn gevonden. Toch was er maar één elementaire verdeling van verantwoordelijkheden. Traditioneel zien we de 'projectorganisatie', 'lijnorganisatie' en 'staforganisatie' (zie figuur 11).

Kort samengevat, richt de projectorganisatie zich op het realiseren van goed gedefinieerde eenmalige doelen, de lijnorganisatie richt zich op de lange termijn doelstellingen zoals onderhoud (termen zoals 'testfabriek' of 'testline' worden gehanteerd in deze context), de staforganisatie ondersteunt in principe mensen in de project- en lijnorganisatie met gespecialiseerde deskundigheid zoals testtooling (dit is ook wel bekend als een Test Expertise Centrum).



Figuur 11. De traditionele verdeling van verantwoordelijkheden in een testorganisatie

In de traditionele IT werd unittesten binnen projecten georganiseerd. Systeemtesten werden dan gedaan door onafhankelijke testteams, zoals een lijnorganisatie (inclusief een uitgebreide regressietest) en de acceptatietest werd gedaan door de business vertegenwoordigers, ondersteund door testprofessionals uit de staforganisatie.

Tegenwoordig is de integratie van al deze activiteiten de trend. Wat betekent dit voor de testorganisatie?

De moderne benadering voor het aanpakken van informatie-technologie uitdagingen is het oprichten van kleine, autonome beslissingsbevoegde teams. De ultieme vorm, bekend als de 'volledige-team-aanpak' of 'DevOps', integreert alle taken vanuit ontwerp, ontwikkeling, onderhoud en beheer. Het onderscheid tussen project- en lijnorganisatie bestaat dan niet meer. Ze hebben zich met elkaar vermengd (zie figuur 12).

Voordelen van deze geïntegreerde organisatie zijn betere communicatie en samenwerking binnen het team, versterking van de verschillende vaardigheden binnen het team in het voordeel van het project, en dat kwaliteit een gedeelde verantwoordelijkheid wordt.

Dit zal heel goed werken in kleine organisaties met slechts een paar teams: alle expertise die effectief de behoeften van de business kan ondersteunen is beschikbaar binnen de teams. Maar wat betekent dit voor grotere organisaties?

Als uw organisatie een groot aantal geïntegreerde teams heeft komt u twee uitdagingen tegen:

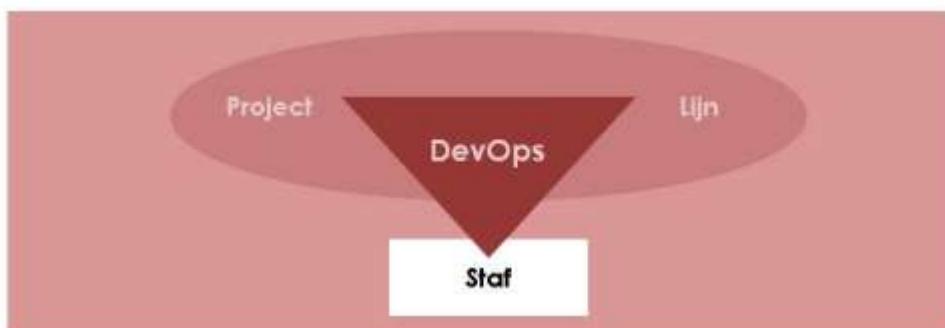
1. Hoe wisselen teams de nodige informatie uit?
2. Hoe verkrijgen de teams vaardigheden en expertise die ze niet in hun team hebben?

Ad 1) Kleine beslissingsbevoegde teams hebben de neiging zich te isoleren, omdat dat afleiding helpt voorkomen, zodat de snelheid erin blijft. Toch moet er informatie uitgewisseld worden om de resultaten van de teams op elkaar af te stemmen. Daarnaast zal op de lange termijn onderhoudbaarheid van de informatiesystemen in een grotere organisatie profiteren van het gebruik van standaards die door de teams onderling worden overeengekomen.

Ad 2) In een klein beslissingsbevoegd team zullen er meestal een of twee mensen zijn die bijzonder bedreven zijn op een bepaald gebied, zoals bijvoorbeeld een business analist, een systeemontwerper, programmeurs, een beheerder of een tester. Op het gebied van testen, zullen de teamleden algemene kennis en vaardigheden hebben, en de tester in het team zal meer diepgaande kennis en vaardigheden daarvan hebben. Echter, één persoon kan geen expert zijn op alle gebieden van het testvakgebied, dus hoe krijgt het team de ontbrekende kennis en vaardigheden?

In een kleine organisatie (laten we zeggen met 3 teams) kent iedereen elkaar en is men in staat om op informele basis de hierboven beschreven uitdagingen te managen.

Maar in grotere organisaties vragen beide uitdagingen om ondersteuning. Een staforganisatie is nodig om deze ondersteuning goed te organiseren. De staforganisatie bestaat uit experts op verschillende gebieden die in staat zijn om meerdere teams in de organisatie te ondersteunen. Zo zal de staforganisatie worden gevraagd om de teams te ondersteunen bij het opzetten van een geautomatiseerde regressietest, of om het totale onderhoud en ondersteuning van de testmanagementtools te doen. Ook zal de staforganisatie de richtlijnen en standaards opstellen waar de teams om vragen (in gedachten houdend dat de staforganisatie niet alleen standaards creëert in het belang van standaardisatie, maar ook op verzoek van de teams om belemmeringen op te lossen).



Afbeelding 12. Testen in geïntegreerde en beslissingsbevoegde teams, ondersteund door de staforganisatie

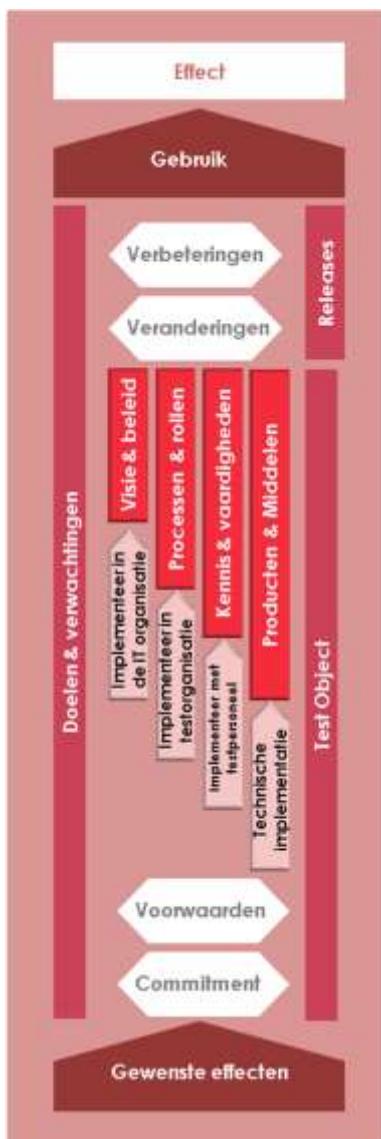
Samenvattend: de belangrijkste reden voor het hebben van een staforganisatie is dat specifieke gespecialiseerde kennis en vaardigheden te schaars zijn binnen de teams en dus moeten worden toegevoegd van buitenaf. Zo verkrijgt u de optimale voordelen van geïntegreerde en beslissingsbevoegde teams.

Integreren is een van de elementen die in dit boek worden geïntroduceerd. De geïntegreerde organisatie is het antwoord op de uitdagingen van vandaag de dag. Op deze manier wordt de zekerheid van voldoende kwaliteit ingebet in de activiteiten van de teams. En wanneer het team zelf bepaalde vaardigheden of deskundigheid ontbeert is de staforganisatie beschikbaar om te helpen.

## 2.19 Building block 19: Implementatie testtools

Om de gewenste effecten van een testtool te bereiken, moeten we die implementeren. Na de implementatie kan het primaire effect meteen bereikt worden, de afgeleide effecten duren langer.

Hoe een testtool geïmplementeerd moet worden varieert per type testtool, maar er zijn generieke aspecten die de implementatie van elk type testtool moet adresseren. Er is meer voor nodig dan alleen het installeren van de tool, zoals wordt gevisualiseerd in het testtool implementatiemodel (zie figuur 13).



Er is altijd een relatie tussen de testtool en het testobject. De meeste vormen van testtools hebben een sterke relatie met de technologie van het testobject. Deze technologie bepaalt of een testtool kan worden gebruikt en zo ja: de hoeveelheid inspanning die nodig is om een bruikbare oplossing te implementeren en te onderhouden. Een andere factor is het aantal releases van het testobject: dit bepaalt de frequentie van het gebruik van de testtool, maar ook de frequentie van het benodigde onderhoud. Het stellen van doelen in termen van scope, resultaten en tijdschema's is een duidelijke best practice, maar het omgaan met verwachtingen is net zo belangrijk: ze kunnen aanzienlijk verschillen van wat er zal worden bereikt.

Het resultaat is dat de implementatie van een testtool vastzit tussen enerzijds de Doelen en Verwachtingen en anderzijds het Testobject en de Releases.

De implementatie begint met het krijgen van commitment en het omgaan met randvoorwaarden. De technische implementatie houdt zich bezig met het installeren en het opzetten van de testtool om een bruikbare en onderhoudbare oplossing te creëren. Een goede technische implementatie is niet genoeg, de mensen blijven de kritische succesfactor en het uitrusten van personeel met de juiste kennis en vaardigheden voor het gebruik en onderhouden van de testtool is essentieel. De implementatie is echt succesvol wanneer het gebruik van de testtool (en natuurlijk de nodige activiteiten om deze te kunnen blijven gebruiken) is uitgegroeid tot een integraal onderdeel van het testproces (of nog beter: het ontwikkelingsproces). Met andere woorden: het gebruik van de testtool is vanzelfsprekend geworden.

Na de implementatie kunnen we de testtool gebruiken,  
maar we moeten ons blijven aanpassen aan

Figuur 13. Testtool Implementatie Model

veranderingen om de beoogde effecten te blijven bereiken. De meest voor de hand liggende veranderingen zijn veranderingen in het testobject dat onderhoud van de testtool nodig maakt, maar ook veranderingen in de organisatie of processen. Continu op zoek zijn naar verbeteringen op alle niveaus helpt ons het effect van het gebruik van testtools te verhogen.

De meeste testtool implementaties worden geïnitieerd op operationeel niveau, wat zorgt voor een bottom-up benadering die zich richt op het gebruik van een enkel type testtool voor het testen van een enkele applicatie. Dit is een effectieve aanpak, vooral omdat commitment gemakkelijker bereikt wordt en doelen gemakkelijker bereikt worden als gevolg van het operationele niveau van de doelen. Maar het inbedden van het gebruik van testtools in de levenscyclus van een applicatie (de Visie & Beleid laag) is moeilijker en wordt vaak vergeten.

Een top-down benadering begint met strategische keuzes die de doelstellingen van de gehele organisatie weerspiegelen, zich bezighoudt met alle soorten van testtools en integraties en de individuele implementaties overkoepelt. Hoewel top-down implementatie langer duurt als gevolg van de grotere omvang, maximaliseert het de effecten van het gebruik van testtools door ze te integreren in de gehele levenscyclus van een applicatie.

## **2.20 Building block 20: Reviewen van requirements**

Het doel van de ontwikkeling van software is kwalitatief goed functionerende eindproducten en diensten. Tijdens de ontwikkeling, of het nu waternal of Agile betreft, worden er tussentijdse producten gemaakt om dat doel te bereiken: business case, requirements, plannen, ontwerpen, enzovoorts. Als deze tussenproducten niet de juiste kwaliteit hebben, zullen ze nooit leiden tot het gewenste resultaat.

Door de kwaliteit van deze tussenproducten vast te stellen kunnen potentieel dure fouten in een vroeg stadium gevonden worden. Hoe eerder een fout wordt gevonden, hoe eenvoudiger en goedkoper deze kan worden hersteld. Het doel moet zijn om fouten op te sporen bij de bron. Naast de kosten- en doorlooptijdreductie is een ander voordeel dat de kloof tussen de verwachte en gerealiseerde kwaliteit kleiner wordt. Het beoordelen van tussenproducten kan op veel manieren worden benoemd, zoals 'toetsen', 'review', 'evaluatie' en 'inspectie'.

Niet alleen worden fouten eerder gevonden, een aantal fouten zijn ook eenvoudiger te vinden dan met het eigenlijke testen: gebreken zoals afwijkingen van standaards, onduidelijkheden en inconsistenties, onvoldoende onderhoudbaarheid, etc.

Tussenproducten kunnen worden vergeleken met:

- Het voorgaande tussenproduct.
- Criteria uit de volgende fase (in checklists).
- Andere tussenproducten op hetzelfde niveau.
- Overeengekomen productstandaards.
- De verwachtingen van de klant.

Er zijn verschillende reviewtechnieken, variërend in doel, formele verantwoordelijkheden en procedure. Omdat niet elk tussenproduct moet worden beoordeeld met dezelfde inspanning kunnen er verschillende technieken worden gekozen per tussenproduct. Voor meer informatie zie TMap NEXT, 2006.

Reviewen is niet een moeilijk op te zetten proces, maar in de praktijk kan het proces vastlopen in praktische problemen. Hier is een (niet uitputtende) lijst van wat gedaan moet worden of vermeden moet worden om dit te overwinnen:

- Ga voorbereid naar review meetings.
- Houd gestructureerde review meetings.
- Niet meer dan 6 personen in een vergadering.
- Laat mensen die te dominant zijn (functies / rollen) niet in een vergadering toe.
- Kritiek op het product, niet op de maker.
- Zorg voor een correcte bevindingen administratie en analyseer de root cause (en metrics).
- Zorg voor ondersteuning.
- Verdeel grote documenten om te voorkomen dat alleen de eerste 20 pagina's goed worden gereviewd.
- Varieer in reviewers om aandachtsverslapping te voorkomen.

In de loop van de tijd kan de intensiteit van de reviews dalen, omdat de lessons-learned zouden moeten resulteren in minder 'major' fouten.

### **In meer detail**

Het belang van goede requirements voor een software project mag niet worden onderschat. Analisten stellen dat maar liefst 71% van de software projecten die mislukken dat doen vanwege de kwaliteit van de business requirements. Requirements markeren de grens tussen wat we zouden willen hebben en wat we gaan bouwen. Ze vertegenwoordigen de behoefte van de business owner van het project. Als er fouten of onduidelijkheden in staan, dan is het hele project in gevaar.

Er zijn vele vormen van requirements, maar het belangrijkste is dat iedereen in een project dat met de requirements omgaat, ervoor moet zorgen dat hij of zij begrijpt wat ermee bedoeld wordt. Er zijn vele manieren om daarvoor te zorgen. Op zijn minst moet iedereen die betrokken is zich de volgende vragen stellen:

- Samenhang: zijn er requirements die elkaar tegenspreken op de een of andere manier?
- Volledigheid: beschrijven de requirements alle attributen van het te implementeren systeem?
- Controleerbaarheid: Is het mogelijk om te controleren of een requirement correct is gebouwd?
- Traceerbaarheid: Is het mogelijk om de status van dit requirement in alle stadia van de ontwikkeling van de software te controleren?
- Atomair: Is dit requirement zo eenvoudig mogelijk (maar niet eenvoudiger)? Een eenvoudige check hiervoor is als het requirement woorden bevat als 'en', 'of' en 'maar'.
- Structuur: Hebben alle requirements dezelfde structuur? Er zijn vele manieren om de requirements vast te leggen (denk aan user stories, requirementsdocumenten, enzovoort), maar het allerbelangrijkste is zo consistent mogelijk te zijn in de manier waarop de requirements worden vastgelegd.
- Haalbaarheid: kan het requirement worden bereikt door de organisatie, gezien de huidige stand van tijd, budget en mogelijkheden?
- Begrijpelijkheid: kan iedereen die betrokken is bij het project begrijpen wat er bedoeld wordt met het requirement?

Er zijn veel meer controles die men kan doen om de kwaliteit van de requirements te monitoren. Zo is het vaak handig om te controleren op "zwakke woorden". Dit zijn woorden

die wanneer ze zich voordoen binnen een requirement, vaak deel zijn van een dubbelzinnigheid, of een signaal vormen dat een requirement onduidelijk zou kunnen zijn. Een lijst van 'zwakke woorden' is te vinden op tmap.net.

## **Hoofdstuk 3 Website**

### **3.1 Inleiding**

Hoofdstuk 3 van dit workbook staat in het teken van testontwerp. De bron die wordt weergegeven in dit hoofdstuk is informatie vanuit de Tmap Suite website ([www.tmap.net](http://www.tmap.net)) over dit onderwerp.

Vanwege het veranderlijke karakter van de website hebben we de informatie voor de certificering in het workbook opgenomen. Het workbook blijft daarmee de bron voor de certificering, maar alle onderwerpen die in het workbook aan bod komen, komen ook online aan bod.

#### **3.1.1. Leeswijzer**

In dit hoofdstuk worden een aantal onderwerpen behandeld. In de eerste twee paragrafen wordt het belang van testontwerp uiteengezet en wordt vervolgens een aantal belangrijke begrippen rondom testontwerp toegelicht. In de paragrafen 3.3 t/m 3.6 worden een aantal groepen van verschillende dekkingsvormen toegelicht.

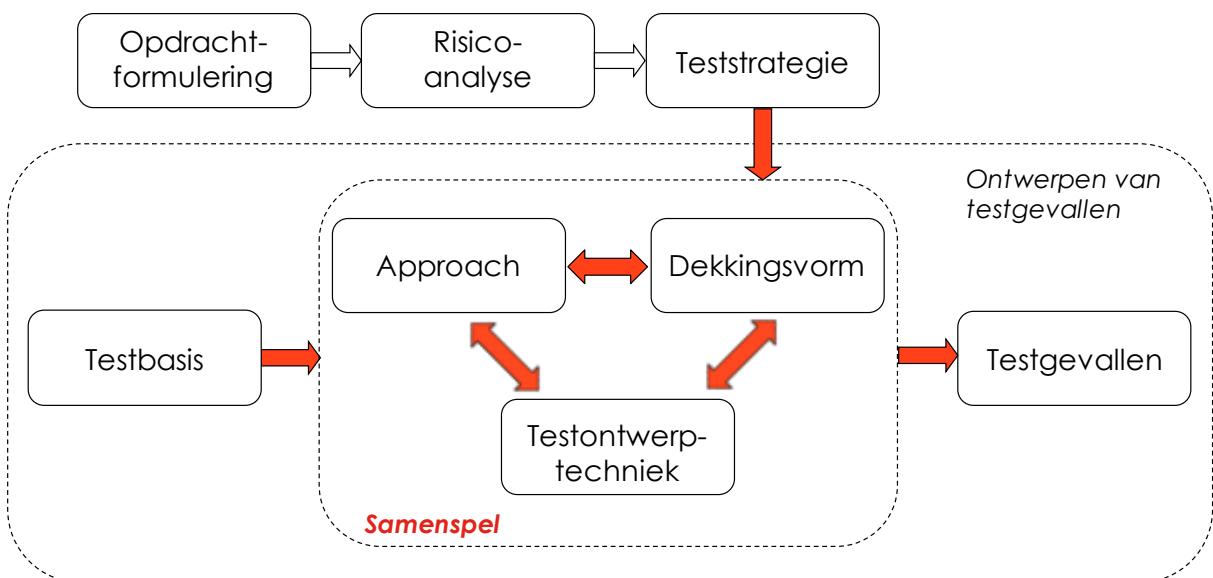
In de laatste paragraaf 3.7 worden een 8-tal testontwerptechnieken toegelicht.

#### **3.1.2. Waarom testontwerp?**

Het doel van testen is het gedegen adviseren over kwaliteit en risico's, zodanig dat alle betrokken partijen vertrouwen in het product krijgen. Om dit te kunnen doen, moet de tester informatie verzamelen over het systeemgedrag. Het belangrijkste middel daartoe is het uitvoeren van testgevallen. De grote vragen zijn nu: "Welke testgevallen? Hoeveel? En hoe komen we aan die testgevallen?" Bij het beantwoorden van die vragen is testontwerp essentieel.

Het bedenken van de juiste (set aan) testgevallen is de essentiële schakel tussen de teststrategie en de concrete testgevallen waarmee deze teststrategie wordt waargemaakt.

Dit vindt plaats in de context 'van testopdracht tot testgevallen' (zie ook Workbook: paragraaf 4.3 "Fase Planning" of op [tmap.net](http://tmap.net): [Test design](#) paragraaf 6.2 "Fase Planning" van TMap NEXT®.)



Figuur 14. Van testopdracht naar testgevallen

De kern hiervan is het maken van keuzes. De rode draad daarbij is (zie figuur 14):

- Het is nooit mogelijk om 'alles' te testen. Bijvoorbeeld vanwege de randvoorwaarden ten aanzien van tijd en kosten die in de opdrachtformulering gesteld zijn. Maar óók omdat vrijwel nooit eenduidig gedefinieerd kan worden wat dat 'alles' nou precies inhoudt (bijvoorbeeld: alle regels code of alle mogelijke combinaties van gegevens of alle kwaliteitsattributen of alle mogelijke paden in het proces of alle fouten of ... of ...?). Er moeten dus allerlei keuzes gemaakt worden.
  - Hoe belangrijker iets is, des te zwaarder het getest moet worden. Deze belangrijkheid (keuze) wordt in kaart gebracht met een risicoanalyse (bijvoorbeeld [PRA](#) of PRBA).
  - In de teststrategie wordt vervolgens een overzicht gemaakt van welke onderdelen hoe zwaar getest gaan worden en hoe de testinspanning verdeeld wordt over de verschillende testvariëteiten (keuzes), zodanig dat de genoemde risico's zo adequaat mogelijk afgedekt worden. De kenmerken van het testobject die beoordeeld moeten worden en de grondigheid waarmee dat moet gebeuren geven samen aan welke dekking behaald moet worden.
  - De teststrategie moet vertaald worden naar testgevallen waarmee de teststrategie ten uitvoer wordt gebracht. In veel gevallen moet de teststrategie bovendien aantoonbaar worden waargemaakt (dit kan bijvoorbeeld als één van de randvoorwaarden in opdrachtformulering zijn vastgelegd).
  - Op welke manier de testgevallen tot stand komen (keuze) hangt van een aantal factoren af. De belangrijkste factoren zijn:
    - De afgesproken dekking (de te testen kenmerken en de zwaarte waarmee dat moet gebeuren)
    - De beschikbare testbasis
    - De manier waarop het systeemontwikkelproces is georganiseerd (bijvoorbeeld agile versus traditioneel)
    - Beschikbare kennis en ervaring
    - Beschikbaar budget (tijd en kosten)
- Op basis van deze factoren wordt een keuze gemaakt – niet in een vaste volgorde – voor test approach(es), dekkingsvormen en testontwerptechnieken.

De keuzevolgorde van Approach(es), Dekkingsvorm(en) en Testontwerptechniek(en) ligt niet op voorhand vast.

Bijvoorbeeld:

In een Agile-omgeving zou eerst voor de Ervaringsgebaseerde approach gekozen kunnen worden en daarbinnen voor Exploratory Testing. Bij het uitvoeren van ET worden vervolgens de meest geschikte dekkingsvormen toegepast (Dekkingsgebaseerde approach).

In een andere situatie wordt bijvoorbeeld op basis van de vereiste dekking en de beschikbare testbasis direct gekozen voor het toepassen van de Procescyclustest, waarbinnen dan nog gekozen wordt voor de testmaat. Hiermee ligt de approach (dekkingsgebaseerd) vast.

In weer een andere situatie wordt eerst een aantal dekkingsvormen gekozen, waarbij de aldus verkregen testsituaties gecombineerd worden tot testgevallen. De specifieke benaming van de testontwerptechniek is dan niet meer relevant (wellicht een volledig nieuwe TOT). Wel ligt ook in dit geval de approach vast: dekkingsgebaseerd.

**Tip:** gebruik bij voorkeur een mix van dekkings- en ervaringsgebaseerde approaches.

- Uiteindelijk leidt dit tot de set testgevallen waarmee in de afgesproken mate kan worden aangetoond dat de afgesproken dekking wordt behaald en die nodig is om de testopdracht goed uit te voeren.

### 3.1.3. Voordelen van testontwerp volgens de TMap Suite

Gedegen testontwerp is dus belangrijk. Behalve het bovenstaande is hiervoor nog een aantal argumenten te noemen:

- Omdat testontwerp, in elk geval in de dekkingsgebaseerde approach (zie Building Block 9 'Test Approaches'), gericht is op het bereiken van een bepaalde dekking voor het vinden van bepaalde typen fouten (bijvoorbeeld in de interfaces, het proces, de invoercontroles of de verwerking), worden dergelijke fouten op een effectieve wijze opgespoord.
- Het testontwerp is er in de meeste gevallen op gericht de vereiste dekking met zo min mogelijk testgevallen te bereiken.
- 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 testontwerp en -uitvoering in goed gedefinieerde blokken kunnen worden opgedeeld.

## **3.2 Kader en begrippen testontwerp**

### **3.2.1. Inleiding**

Bij testontwerp gaat erom te komen tot een set testgevallen waarmee in de afgesproken mate kan worden aangetoond dat de afgesproken dekking wordt behaald.  
Als eerste wordt daarom ingegaan op wat een testgeval nou eigenlijk is.

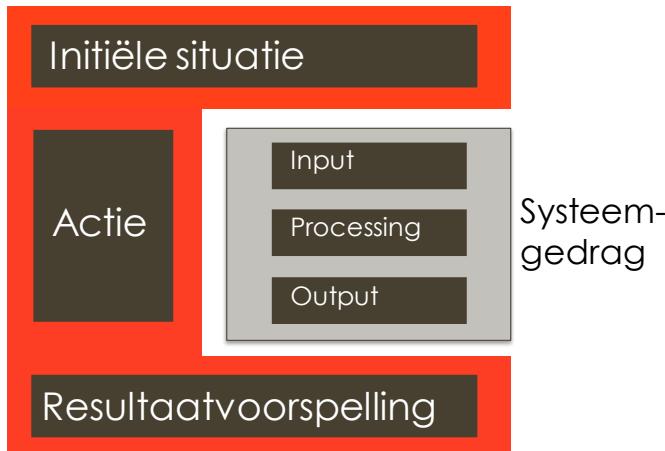
#### **3.2.1.1 Wat is een testgeval?**

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'.

Een testgeval bestaat uit een beschrijving van het startpunt (ook wel initiële situatie genoemd), de uit te voeren test acties en een resultaatvoorspelling:

- Startpunt (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?"

Zie onderstaande figuur 15 voor de generieke opbouw van een testgeval, in relatie tot het te testen systeemgedrag.



Figuur 15. 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 (een te testen mogelijkheid) – is een testgeval een afgerond geheel dat als afzonderlijke test uitgevoerd kan worden.

### 3.2.1.2 Sleutelbegrippen bij testontwerp

De sleutelbegrippen bij testontwerp zijn:

- [Dekking](#)
  - [Dekkingsvorm](#)
  - [Testzwaarte](#)
- [Test approach](#)
- [Testontwerptechniek](#)

Deze begrippen en hun onderlinge samenhang worden in de paragrafen 3.2.3 tot en met 3.2.5 toegelicht. Eerst wordt in paragraaf 3.2.2 uitgelegd uit welke generieke stappen testontwerp bestaat en hoe dekkingsvormen en testontwerptechnieken met elkaar samenhangen.

### 3.2.2. De generieke testontwerpstappen

Bij het testontwerp worden de volgende vijf generieke stappen doorlopen:

1. Identificeren testsituaties.
2. Opstellen logische test gevallen.
3. Opstellen fysieke test gevallen.
4. Vaststellen uitgangssituatie.
5. Opstellen testschrift.

Deze vijf stappen gelden in essentie *altijd*, ongeacht de gekozen testontwerptechniek (of dekkingsvorm of test approach). Niet in alle gevallen wordt iedere stap expliciet doorlopen en soms worden stappen samengevoegd. Dat kan te maken hebben met de specifiek toegepaste techniek, maar ook met praktische overwegingen.

De relatie tussen deze stappen is weergegeven in figuur 16:



Figuur 16. Relaties tussen de begrippen testsituaties- testgevallen – testscripts.

De figuur kan worden samengevat als:

- Iedere testsituatie wordt door minimaal één logisch testgeval afgedekt.
- Een logisch testgeval bestaat uit één of meer testsituaties.
- Elk logisch testgeval wordt concreet uitgewerkt in precies één fysiek testgeval.
- Elk fysiek testgeval komt in één testscript voor.

De figuur laat ook het onderscheid zien tussen het logisch en fysiek testontwerp:

- Het *logisch testontwerp* bestaat uit de testsituaties en de logische testgevallen. Hiermee wordt aangetoond dat de afgesproken dekking wordt behaald en dat de teststrategie dus wordt waargemaakt.
- Het *fysiek testontwerp* bestaat uit de concreet uitgewerkte fysieke testgevallen, vastgelegd in testscripts. Dit zorgt voor een gedegen voorbereiding van de testuitvoering. Het fysiek maken van testgevallen voegt dus niets toe aan de zwaarte van de test.

### De begrippen nader toegelicht

- **Stap 1** van het testontwerp is de identificatie van testsituaties.  
Een **testsituatie** is:

Een geïsoleerde omstandigheid (mogelijkheid) die getest moet worden.

In een dekkingsgebaseerde approach worden de testsituaties per definitie verkregen door één of meerdere **dekkingsvormen** toe te passen. In een ervaringsgebaseerde approach worden de testsituaties gebaseerd op vaardigheden, intuïtie en ervaring van de tester.

- In stap 2 en 3 worden de testgevallen vastgesteld.  
Met een **testgeval**:

wordt onderzocht of het systeem **onder bepaalde omstandigheden** (de testsituaties) het gewenste gedrag vertoont.

Een testgeval loopt van 'begin' (invoer) tot 'eind' (uitvoer) en omvat één of meer testsituaties.

- **Stap 2** houdt in dat de testsituaties gecombineerd worden tot logische test gevallen, zodanig dat elke testsituatie door minimaal één logisch testgeval wordt afgedekt.  
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

In andere woorden: datgene wat getest gaat worden, in abstracte termen aangegeven.

- **Stap 3** houdt in dat de logische test gevallen voldoende concreet worden uitgewerkt om de test gevallen ook daadwerkelijk uit te kunnen voeren. Een worden keuzes gemaakt ten aanzien van de fysieke waardes.

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

Fysieke test gevallen bevatten daarbij meestal een concrete beschrijving van:

- Initiële situatie
  - Alles wat nodig is om systeeminput te kunnen ontvangen, zoals:
    - Database met benodigde gegevens
    - Omgevingsparameters, bijv. systeemdatum
    - Toestand van het systeem
- Actie
  - Alle activiteiten nodig om systeemgedrag te activeren:
    - Simpel: runnen batchprogramma of invoeren gegevens
    - Complex: heel veel acties
- Resultaatvoorspelling
  - Alle resultaten die gecontroleerd moeten gaan worden, zoals:
    - Juiste boodschap op scherm
    - Database wel/niet aangepast
- **Stap 4** omvat het vaststellen van de uitgangssituatie, oftewel alles wat nodig is om de test gevallen uit te kunnen voeren. De uitgangssituatie voor een testontwerp omvat de initiële situaties van de afzonderlijke test gevallen uit het testontwerp, aangevuld met alles wat verder nog nodig is om de set aan test gevallen te kunnen uitvoeren. De uitgangssituatie wordt voorafgaand aan de test uitvoering 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.

- **Stap 5** is het opstellen van het testscript. Hierbij zijn de test-acties en controles van de fysieke test gevallen beschreven in een voor test uitvoering optimale volgorde. Hierbij moeten de test gevallen elkaar niet kunnen verstören. Het testscript is als zodanig het stappenplan voor de test uitvoering en biedt tevens de mogelijkheid van voortgangsbewaking. De fysieke test gevallen en de uitgangssituatie vormen uiteraard de basis voor het te vervaardigen testscript.

De generieke inhoudsopgave van een script is als volgt:

- Unieke identificatie, bestaande uit:

- o versie;
- o opsteller;
- o 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).

### **3.2.3. Dekking, dekkingsvormen en testzwaarte**

#### **3.2.3.1 Dekking**

De keuzes in je teststrategie over WAT te testen geeft aan dat je bepaalde aspecten van het testobject wil afdekken. Het doel van een effectieve teststrategie is dan ook om de best haalbare dekking te realiseren op de juiste plaats. Dekking heeft alles te maken met de wens om de grootst mogelijke hoeveelheid fouten te vinden met zo min mogelijk testcases.

Maar wat is de dekking? Dekking is zeer subjectief. We kunnen niet spreken over DE dekking. Wat bedoelt een opdrachtgever of een andere stakeholder als hij / zij je vraagt wat de dekking van de test was? Welke informatie heeft hij / zij nodig? Mogelijk wil hij / zij weten hoe grondig sommige aspecten van het testobject zijn gedekt. Misschien wil hij / zij weten hoeveel van alle mogelijke fouten daadwerkelijk zijn gevonden door de tests.

Een sleutelwoord hier is dekking. Een definitie van dekking is moeilijk. Het gaat in principe om de aspecten van het testobject die je wil beoordelen en de grondigheid waarmee je dat doet.

Belangrijker is de vraag of we in staat zijn om 100% dekking te bereiken. We kunnen nooit zeker van zijn dat alle fouten zijn gevonden of zelfs dat 60% van alle fouten is gevonden. Immers, we weten niet hoeveel fouten er eigenlijk zijn. Verder weten we niet hoe nauwkeurig en volledig de informatie was waarop onze testgevallen zijn gebaseerd. Ook als we de tests uitgevoerd hebben op basis van een teststrategie (en productrisicoanalyse) kunnen we nooit zeker zijn of onze stakeholders de juiste keuzes hebben gemaakt over de testdekking. Alles testen is simpelweg onmogelijk, omdat het onmogelijk is te bepalen wat we onder *alles* verstaan.

Hoewel dekking kortom moeilijk te definiëren is, gaat het om:

- de aspecten (bijvoorbeeld kwaliteitsattributen) van het testobject die beoordeeld moeten worden
- én
- de grondigheid waarmee dat moet gebeuren

### 3.2.3.2 Dekkingsvormen

Definitie:

Dekkingsvorm

De vorm waarin testsituaties zijn af te leiden uit de testbasis.

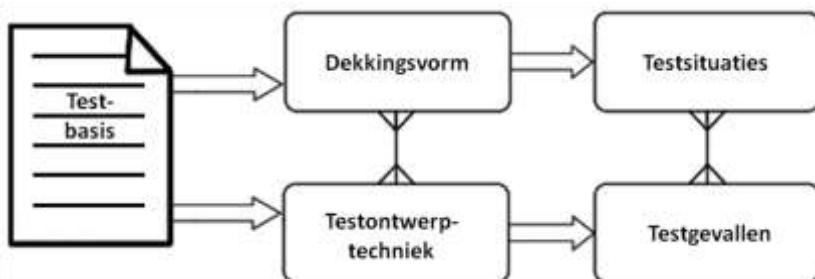
Dit betreft:

- de mogelijkheden die getest moeten worden
- én de werkwijze om die mogelijkheden te identificeren.

Een dekkingsvorm is gericht op het bereiken van een specifieke dekking om specifieke soorten fouten op te sporen (bijvoorbeeld in de interfaces, controles op de input of de verwerking). Op deze manier worden dergelijke fouten effectiever gedetecteerd dan door het specificeren van ad hoc testgevallen. Een dekkingsvorm kan in de praktijk alleen 'beter' worden genoemd als het ten minste alle fouten vindt die door een andere dekkingsvorm zouden worden gevonden, plus een aantal extra fouten.

Samengevat (zie figuur 17):

- Het is niet mogelijk om alles te testen binnen de grenzen van de randvoorwaarden tijd en kosten, zoals bepaald in de opdracht. Keuzes moeten worden gemaakt met betrekking tot de mate waarin getest wordt.
- Een teststrategie wordt gebruikt om overzicht te krijgen wat er wordt getest en hoe grondig, zodat de te testen aspecten zo goed mogelijk worden afgedekt.
- De beslissingen over grondig en minder grondig testen worden vertaald naar concrete uitspraken over de beoogde dekking.
- Afhankelijk van onder andere de beschikbare testbasis worden passende dekkingsvormen geselecteerd om de dekking te bereiken.



Figuur 17. Afleiden van testsituaties vanuit dekkingsvorm .

### 3.2.3.3 Testzwaarte

In de teststrategie is afgesproken welke testzwaarte moet worden bereikt. Samen met de aspecten van het testobject die beoordeeld moeten worden geeft dit de te bereiken dekking weer.

Opdrachtgevers en andere stakeholders willen waarschijnlijk informatie over de daadwerkelijk gerealiseerde zwaarte van de test. Maar wat bedoelen ze? Voor testzwaarte is geen eenduidige definitie te geven. Het gaat hierbij om vragen als:

- Hoe grondig was de gekozen dekkingsvorm?
- Werden meerdere dekkingsvormen toegepast?
- Is er binnen de specifieke dekkingsvorm gevarieerd in grondigheid?

Er is géén sprake van 'beter'!

Hoewel een fascinerend onderwerp, is het ook een complexe materie. Er is geen zwart en wit. Echter, we kunnen stellen dat alles testen onmogelijk is. Hoeveel grondiger is een bepaalde dekkingsvorm in vergelijking tot andere dekkingsvormen (bijvoorbeeld Pairwise testing versus Modified Condition/Decision coverage)? Hoeveel grondiger is een variatie binnen een dekkingsvorm in vergelijking tot een andere variatie binnen dezelfde dekkingsvorm (bijvoorbeeld Modified Condition/Decision coverage versus Modified Condition coverage)? Hoeveel extra fouten zijn te vinden?

### 3.2.3.4 Dekkingsgroepen

Dekkingsvormen kunnen worden onderverdeeld in vier dekkingsgroepen:

**Proces:** Processen kunnen worden geïdentificeerd op verschillende niveaus. Zo zijn er algoritmes voor control flows en bedrijfsprocessen. Dekkingsvormen zoals Paden, Statement coverage en State transitions kunnen worden gebruikt om (variaties in) deze processen te testen.

**Condities:** In bijna elk systeem zijn er beslispunten, bestaande uit voorwaarden (condities), waar het systeemgedrag in verschillende richtingen kan gaan, afhankelijk van de uitkomst van een dergelijk beslispunt. Variaties in deze condities en hun uitkomsten kunnen worden getest met dekkingsvormen als onder andere Decision coverage, Modified Condition/Decision coverage en Multiple Condition coverage.

**Gegevens:** Gegevens worden aangemaakt en eindigen wanneer deze worden verwijderd. In de tussentijd worden de gegevens gebruikt door deze bij te werken of te raadplegen. Deze levenscyclus van gegevens kan worden getest, maar ook de combinaties van invoergegevens, evenals de rubrieken van invoer- of uitvoergegevens. Voorbeelden van dekkingsvormen hierbij zijn Grenswaardenanalyse, CRUD, Gegevensstromen en Syntactische test.

**Voorkomen:** Hoe een systeem werkt, hoe het presteert, wat het uiterlijk zou moeten zijn, wordt vaak beschreven in niet-functionele requirements. Binnen deze groep vinden we dekkingsvormen zoals Operational profiles, Load profiles en Presentatie.

### 3.2.3.5 Dekkingsvormen per dekkingsgroep

In onderstaande tabel wordt per groep een korte beschrijving gegeven van de daarbij behorende dekkingsvormen.

NB. In de volgende paragraaf wordt aangegeven hoe de testzwaarte gevarieerd kan worden bij verschillende dekkingsvormen.

GROEP	DEKKINGSVORM	BESCHRIJVING
Proces	<u>Control flow</u>	Testen van de structuur van een programma.
	<u>Paden</u>	Dekken van de variaties in het procesverloop in termen van combinaties van paden. Als testbasis is een schema nodig van beslispunten en paden.
	Zeldzame gebeurtenissen	Adresseren van gebeurtenissen die niet vaak voorkomen.

GROEP	DEKKINGSVORM	BESCHRIJVING
	Goedpaden/Foutpaden	Dekken van het goedpad en het foutpad bij iedere gedefinieerde foutsituatie. Een ongeldige situatie (foutieve controles in het proces of algoritme die voorafgaan aan de verwerking) moet leiden tot correcte foutafhandeling, terwijl een geldige situatie moet worden geaccepteerd door het systeem zonder foutafhandeling.
	State transitions	Verificatie van relaties tussen gebeurtenissen, acties, activiteiten, statussen en statusovergangen.
Condities	<u>Beslispunten</u>	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.
	<u>Semantiek</u>	Validatie van relaties tussen gegevens.
Gegevens	<u>Grenswaardenanalyse</u>	Een grenswaarde bepaalt de overgang van de ene equivalentieklas naar de andere. Grenswaardenanalyse test de grenswaarde zelf plus de waarde direct hierboven en direct hieronder.
	<u>CRUD</u>	Dekken van alle basisbewerkingen (Create, Read, Update, Delete) op alle entiteiten.
	<u>Gegevenscombinaties</u>	Testen van combinaties van parameter waarden. Basis zijn de <u>Equivalentieklassen</u> .
	Gegevensstromen	Verificatie van informatie van een gegevensstroom, die loopt van actor tot actor, van invoer tot uitvoer.
	Domain testing	Dekking van een klein aantal waarden uit een bijna oneindige groep van kandidaat-waarden. Hierbij is domeinkennis van cruciaal belang.

GROEP	DEKKINGSVORM	BESCHRIJVING
Onderliggende dekkingsvormen	<u>Equivalentieklassen</u>	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.
	<u>Integriteitregels</u>	Het controleren van de voorwaarden onder welke bepaalde CRUD processen wel of niet zijn toegestaan.
	Goedpaden/Foutpaden	Dekken van het goedpad en het foutpad bij iedere gedefinieerde foutsituatie. Een ongeldige situatie (bepaalde waarden of combinaties van waarden die zijn gedefinieerd en die niet zijn toegestaan voor de desbetreffende functionaliteit) moet leiden tot correcte foutafhandeling, terwijl een geldige situatie moet worden geaccepteerd door het systeem zonder foutafhandeling.
	<u>Syntax</u>	Validatie van rubrieken van invoer- of uitvoergegevens.
Voorkomen	<u>Heuristiek</u>	Evaluatie van (een aantal) bruikbaarheid beginselen.
	<u>Load profiles</u>	Simuleren van een realistische belasting van het systeem in termen van aantallen gebruikers en/of transacties.
	<u>Operational profiles</u>	Simuleren van het realistisch gebruik van het systeem, door een opeenvolging van transacties uit te voeren, die statistisch verantwoord is samengesteld.
	<u>Presentatie</u>	Het testen van de lay-out van invoer (schermen) en uitvoer (lijsten, rapporten).

### 3.2.3.6 Variaties in dekkingsvormen

De beslissing om 'grondiger te testen' kan in principe op 3 manieren vorm worden gegeven door te variëren met dekkingsvormen:

- een grondiger dekkingsvorm;

- meerdere dekkingsvormen;
- een hogere mate van grondigheid binnen een specifiek dekkingsvorm.

Bij een aantal dekkingsvormen is het mogelijk om binnen de dekkingsvorm te variëren in de testwaarde. Onderstaande tabel geeft hiervan een aantal voorbeelden.

DEKKINGSVORM	VARIATIE
<u>Control flow</u>	<ul style="list-style-type: none"> <li>• Statement coverage</li> <li>• <u>Decision coverage</u> (branch testing / arc testing)</li> <li>• <u>Paden</u></li> </ul>
<u>Paden</u>	testmaat-N
State transitions	<ul style="list-style-type: none"> <li>• 0-switch</li> <li>• 1-switch</li> <li>• 2-switch</li> </ul>
<u>Beslispunten</u>	<ul style="list-style-type: none"> <li>• <u>Condition coverage</u></li> <li>• <u>Decision coverage</u></li> <li>• <u>Condition/Decision coverage</u></li> <li>• <u>Modified Condition/Decision coverage</u></li> <li>• <u>Multiple Condition coverage</u></li> <li>• Cause Effect Graph</li> <li>• Pairwise testing</li> </ul>
<u>Semantiek</u>	Zie <u>Beslispunten</u> en <u>Equivalentieklassen</u>
<u>Grenswaardenanalyse</u>	<ul style="list-style-type: none"> <li>• Licht (grenswaarde + één waarde rondom grens)</li> <li>• Normaal (grenswaarde + twee waarden rondom grens)</li> </ul>
<u>Gegevens</u>	<ul style="list-style-type: none"> <li>• Goedpaden/Foutpaden</li> <li>• Geen "data pairs"</li> <li>• Eén of meerdere "data pairs"</li> <li>• N-wise (uitbreiding op "pairwise")</li> <li>• Alle mogelijke combinaties</li> </ul>
<u>Integriteitregels</u>	Zie <u>Beslispunten</u> en <u>CRUD</u>
<u>Syntax</u>	Zie de individuele testsituaties

### **3.2.4. Test approaches**

In aanvulling op hetgeen reeds in paragraaf 2.9 (Test Approaches) is weergegeven, in deze paragraaf nog een aantal aandachtspunten.

Definitie:

Test approach

Een benadering voor het maken van testgevallen.

Er zijn twee benaderingen:

1. Dekkingsgebaseerde approach
2. Ervaringsgebaseerde approach
  - Gebruik bij voorkeur een mix van dekkings- en ervaringsgebaseerde approaches.
  - Dekkingsgebaseerde approach:
    - Testsituaties worden m.b.v. dekkingsvormen uit de testbasis afgeleid;
    - Gericht op effectief en efficiënt verzamelen van informatie over kwaliteit en risico's;
    - Gericht op het aantoonbaar realiseren van de in de teststrategie afgesproken dekking.
  - Ervaringsgebaseerde approach:
    - Laat de tester vrij om testgevallen voorafgaand aan en/of tijdens de testuitvoering te ontwerpen / bedenken;
    - Gebaseerd op vaardigheden, *intuïtie* en ervaring van de tester;
    - Ook gericht op het waarmaken van de teststrategie, maar verschafft minder zekerheid over de gerealiseerde dekking;
    - Dekking moeilijker (of niet) aantoonbaar;
    - Altijd een waardevolle aanvulling op dekkingsgebaseerde approach.

### **3.2.5. Testontwerptechnieken**

Definitie:

Testontwerptechniek

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

Het belang van het toepassen van testontwerptechnieken wordt door onderstaande argumenten weergegeven:

- 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.

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.

De generieke stappen van testontwerp, en dus van het toepassen van testontwerptechnieken zijn in paragraaf 3.2.2 beschreven.

### 3.2.5.1 Relatie tussen dekkingsvorm en testontwerptechnieken

Een testontwerptechniek wordt gebruikt om de benodigde testgevallen af te leiden om de vereiste dekking van een specifieke testbasis te bereiken. De eerste stap van een testontwerptechniek is de identificatie van testsituaties. De testsituaties worden verkregen door toepassing van **dekkingsvormen**. Een testontwerptechniek houdt in dat één of meer dekkingsvormen worden toegepast en geeft aan hoe de verkregen testsituaties tot testgevallen worden samengebracht, zodanig dat elke testsituatie door minimaal één testgeval wordt afgedekt.

De benodigde dekking wordt concreet uitgedrukt in de geselecteerde dekkingsvormen. Elke dekkingsvorm gebruikt een specifiek type informatie in de testbasis, bijvoorbeeld een gestructureerd stroomdiagram met paden en beslispunten.

### 3.2.6. Selectie van dekkingsvormen en testontwerptechnieken

Er bestaan vele soorten van dekkingsvormen en testontwerptechnieken. Omwille van de eenvoud en bruikbaarheid behandelen we hier alleen de meest gebruikte testontwerptechnieken en toepassing van de onderliggende dekkingsvormen.

In onderstaande tabel worden per dekkingsgroep de meest voorkomende dekkingsvormen genoemd, inclusief de testontwerptechnieken waarin ze kunnen worden toegepast.

GROEP	TESTZWAARTE: LICHT	TESTZWAARTE: GEMIDDELD	TESTZWAARTE: ZWAAR
Conditie	<u>Condition/Decision coverage</u> – <u>Elementaire vergelijkingentest</u>	<u>Modified Condition/Decision coverage</u> – <u>Elementaire vergelijkingentest</u> of <u>Condition/Decision coverage</u> – <u>Beslistabeltest</u>	<u>Multiple Condition coverage</u> – <u>Elementaire vergelijkingentest</u> of <u>Multiple Condition coverage</u> – <u>Beslistabeltest</u>
Gegevens	Eén of meerdere data pairs – <u>Datacombinatietest</u>	Pairwise – <u>Datacombinatietest</u>	N-wise of alle combinaties – <u>Datacombinatietest</u>
Proces	Statement coverage en <u>Paden</u> testmaat-1 – <u>Procescyclustest</u>	<u>Decision coverage</u> en <u>Paden</u> testmaat-2 – <u>Procescyclustest</u>	<u>Paden</u> testmaat-2 - <u>Algoritmetest</u> en <u>Paden</u> testmaat-3

GROEP	TESTZWAARTE: LICHT	TESTZWAARTE: GEMIDDELD	TESTZWAARTE: ZWAAR
			- Procescyclustest

NB. De dekkingsgroep "Voorkomen" is hierbij buiten beschouwing gelaten. In de gevallen waarin die dekkingsgroep van toepassing is, is de dekkingsvorm en de testdiepte teveel afhankelijk van de specifieke situatie en het resultaat dat de tester wil bereiken.

### 3.3 Dekkingsvormen Proces

#### 3.3.1. Inleiding

Processen kunnen worden geïdentificeerd op verschillende niveaus. Zo zijn er algoritmes voor control flows en bedrijfsprocessen. Dekkingsvormen uit deze groep kunnen worden gebruikt om (variaties in) deze processen te testen.

Deze groep bestaat uit de volgende dekkingsvormen:

- Paden (paragraaf 3.3.2)
- Control Flow (wordt niet explicet in dit Workbook uitgelegd)
- Goedpaden / Foutpaden (wordt niet explicet in dit Workbook uitgelegd)
- State Transitions (wordt niet explicet in dit Workbook uitgelegd)
- Zeldzame gebeurtenissen (wordt niet explicet in dit Workbook uitgelegd)

#### 3.3.2. Paden

##### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Proces
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Functionaliteit</li> <li>• Inpasbaarheid <ul style="list-style-type: none"> <li>• Werkprocedures in bedrijfprocessen</li> <li>• Workflow</li> </ul> </li> <li>• Structuur van code</li> <li>• Beveiliging</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• Schema van beslispunten en paden</li> </ul>

## Beschrijving

Het dekken van paden is van toepassing als het gedrag van het systeem beschreven is met behulp van beslispunten en paden. Het getoonde figuur laat hiervan een voorbeeld zien.

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** (binnen dekkingsvorm Paden ook wel **padcombinaties** genoemd) 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 dekkingsvorm toepasbaar voor meerdere testvariëteiten.

Het abstractieniveau doet niet ter zake: dekkingsvorm Paden is toepasbaar op zowel gedetailleerd niveau (code algoritme) als op overkoepelend systeem- of bedrijfsprocesniveau, zolang de informatie over het gewenste systeemgedrag maar in de gestructureerde vorm van beslispunten en paden gegeven is.

### Lichtere of zwaardere dekking: de testmaat

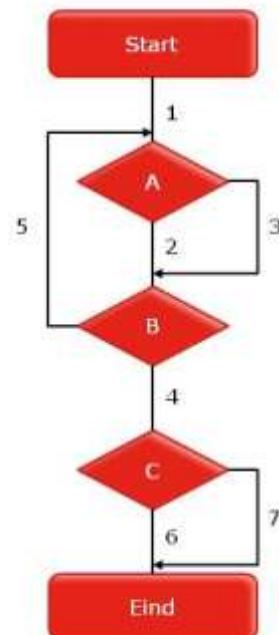
Bij het dekken van paden zijn verschillende zwaarteniveaus mogelijk. Hoe hoger het zwaarte-niveau, hoe groter de foutvindkans. Dit wordt hieronder toegelicht.

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. Door het proces van "Start" tot "Eind" te doorlopen, waarbij elk afzonderlijk pad minimaal een keer wordt geraakt, 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.

Om dit soort fouten te vinden, moet zwaarder getest worden. De testzwaarte wordt bij dekkingsvorm Paden 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 de 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: ieder afzonderlijk pad is een testsituatie. In het voorbeeld zijn dat de paden 1 tot en met 7.

De variatie van dekkingsvormen voor het behalen van testmaat-2 wordt direct hieronder beschreven. Daarna wordt toegelicht hoe hogere testmaten eenvoudig afgeleid kunnen worden vanuit testmaat-2.

### **3.3.2.1 Afleiden van testsituaties voor 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:

- Beslispunten & Paden.

In het processchema de beslispunten benoemen (A, B, enz.) en de paden nummeren. Per beslispunt opsommen van:

- a. Inkomende paden ("IN")
- b. Uitgaande paden ("UIT")

- 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.

Toegepast op het voorbeeld:

<b>Beslispunt</b>	<b>IN</b>	<b>UIT</b>	<b>Testsituaties (padcombinaties)</b>
A	1, 5	2, 3	1-2; 1-3; 5-2; 5-3
B	2, 3	4, 5	2-4; 2-5; 3-4; 3-5
C	4	6, 7	4-6; 4-7

### **3.3.2.2 Afleiden van testsituaties voor hogere testmaten**

Voor hogere testmaten wordt het volgende 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.

Het zou geformuleerd kunnen worden als:

Testmaat-(N+1) = Testmaat-N + "1 stap verder in het procesverloop".

Voor het voorbeeld kan dit voor testmaat-3 als volgt worden uitgewerkt:

Uit het schema is eenvoudig het volgende af te leiden:

- Pad 1 is het startpunt voor ieder testgeval;
- Paden 2 en 3 worden gevuld door paden 4 en 5;
- Pad 5 wordt gevuld door paden 2 en 3;
- Pad 4 wordt gevuld 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.

De manieren waarop de met dekkingsvorm Paden verkregen testsituaties gecombineerd kunnen worden tot logische testgevallen, worden beschreven in paragraaf 3.7.3 ([Procescyclustest](#)).

## 3.4 Dekkingsvormen Condities

### 3.4.1. Inleiding

Bij vrijwel ieder systeem is sprake van beslispunten, bestaande uit condities, waar het systeemgedrag verschillende kanten op kan, afhankelijk van de uitkomst van zo'n beslispunt.

Variaties van dergelijke condities en de bijbehorende uitkomsten kunnen worden getest door gebruik te maken van verschillende dekkingsvormen, zoals hieronder aangegeven. Dekkingsvormen binnen deze groep zijn:

- Beslispunten (paragraaf 3.4.2)
  - Condition coverage
  - Decision coverage
  - Condition/Decision coverage
  - Modified Condition/Decision coverage
  - Multiple Decision coverage
- Semantiek (paragraaf 3.4.3)

### 3.4.2. Beslispunten

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Condities
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Functionaliteit</li> <li>• Beveiliging</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• Functioneel ontwerp</li> </ul>
<b>5 dekkingsvormen</b>	<ul style="list-style-type: none"> <li>• Condition Coverage (CC)</li> <li>• Decision Coverage (DC)</li> <li>• Condition / Decision Coverage (CDC)</li> <li>• Modified Condition / Decision Coverage (MCDC)</li> <li>• Multiple Condition Coverage (MCC)</li> </ul>

## Beschrijving

Bij vrijwel ieder systeem is sprake van beslispunten, waar het systeemgedrag verschillende kanten op kan, afhankelijk van de uitkomst van zo'n beslispunt.

Definitie:

Beslispunt

Een beslispunt is een samenstelling van één of meer condities die de voorwaarden definiert voor de verschillende mogelijkheden in het daaropvolgende systeemgedrag.

De verschillende condities bepalen gezamenlijk de uitkomst 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 propositiologica – 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	$\leq 8$
Resultaat	<b>waar (1)</b>	<b>onwaar (0)</b>

Beslispunten kunnen ook bestaan uit combinaties van condities, de zogenaamde **samengestelde conditie**. Vergelijk de volgende samengestelde condities:

ALS ( aantal boeken > 8 OF bedrag  $\geq €250$  ) DAN extra korting en

ALS ( aantal boeken > 8 EN bedrag  $\geq €250$  ) 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.

Deze tabel laat de uitkomsten van alle situaties van een volledige beslistabel zien. Deze tabel wordt een waarheidstabell genoemd. Dit is een veel gebruikte manier

**Waarheidstabell**

A	B	A OF B
1	1	1
1	0	1
0	1	0
0	0	0

**Waarheidstabell**

A	B	A EN B
1	1	1
1	0	0
0	1	0
0	0	0

Met de operatator "OF" is het eindresultaat alleen onwaar als beide condities onwaar zijn.

Met de operator "EN: is het eindresultaat alleen **waar** als beide condities waar zijn

Uitgaande van twee condities levert dit de volgende mogelijke combinaties op. Deze tabel met combinaties wordt wel de **volledige beslistabel** genoemd

**Volledige beslistabel**

A	B
1	1
1	0
0	1
0	0

Het uiteindelijke resultaat in elk van de vier situaties hangt af van de operator "EN" of "OF"

**Waardeidstabel**

A	B	C	(A OF B) EN C
1	1	1	1
1	1	0	0
1	0	1	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

Binnen een samengestelde conditie kan sprake zijn van verschillende operatoren.  
Bij afwezigheid van de haakjes geldt de regel dat de EN voor de OF gaat

### 3.4.2.1 Condition coverage

Met Condition coverage worden de mogelijke uitkomsten ("waar" of "onwaar") van elke conditie minimaal één keer getest. Dit betekent dat elke afzonderlijke conditie één keer waar en één keer onwaar is. Met andere woorden: we dekken alle condities, vandaar de term Condition coverage.

De uitkomst van het beslispunt is alleen van belang voor het controleren van de condities. Hierbij zijn de combinaties van de condities niet relevant. Omdat er slechts twee mogelijke uitkomsten van een conditie zijn (waar of onwaar) zal Condition coverage resulteren in 2 testsituaties per beslispunt.

In de praktijk wordt deze dekkingsvorm daarom niet vaak gebruikt voor het testen van de combinaties van de condities, maar wel wanneer men de uitkomst van het beslispunt belangrijk vindt.

**Waardeidstabel**

A	B	A OF B
1	1	1
1	0	1
0	1	0
0	0	0

Condition Coverage  
De mogelijke uitkomsten ("waar" of "onwaar") van elke condities wordt minimaal één keer getest

Als aantal boeken > 8 OF bedrag  $\geq 100$  DAN Extra korting

	Aantal boeken > 8	Bedrag $\geq 100$	Uitkomst
TS1	1	0	1 (extra korting)
TS2	0	1	1 (extra korting)

De conditie Aantal Boeken > 8 is één keer WAAR en één keer ONWAAR

Ook de conditie Bedrag  $\geq 100$  is één keer WAAR en één keer ONWAAR

Voor Condition coverage zijn er maar twee testsituaties per beslispunt nodig.

### 3.4.2.2 Decision coverage

Met Decision coverage worden de mogelijke uitkomsten van de beslissing minimaal één keer getest. Dit betekent dat de uitkomst van het beslispunt één keer waar en één keer onwaar is. Met andere woorden: we dekken één keer de DAN-tak en één keer de ANDERS-tak.

Hierbij is het van belang om te variëren in de uitkomst van het beslispunt, maar niet noodzakelijkerwijs in die van de condities. Omdat er slechts twee mogelijke uitkomsten van een beslispunt zijn (DAN of ANDERS) zal Decision Coverage resulteren in 2 testsituaties per beslispunt.

**Waardeidstabel**

A	B	A OF B
1	1	1
1	0	1
0	1	0
0	0	0

Decision Coverage  
De mogelijke uitkomsten ("waar" of "onwaar") van de beslissing worden minimaal één keer getest

Als aantal boeken > 8 OF bedrag  $\geq 100$  DAN Extra korting

	Aantal boeken > 8	Bedrag $\geq 100$	Uitkomst
TS1	0	1	1 (extra korting)
TS2	0	0	0 (geen korting)

De uitkomst van de individuele condities hoeft niet te verschillen

De uitkomst van beslissing is één keer WAAR en één keer ONWAAR

Voor Decision coverage zijn er maar twee testsituaties per beslispunt nodig.

### 3.4.2.3 Condition/Decision coverage

Met Condition/Decision coverage worden de mogelijke uitkomsten van elke conditie één van de beslissing minimaal één keer getest. Dit impliceert zowel Condition coverage als Decision coverage. Met andere woorden: we zorgen dat alle condities één keer waar en één keer onwaar zijn EN we dekken één keer de DAN-tak en één keer de ANDERS-tak.

### Waardeidstabel

A	B	A OF B	Condition/Decision Coverage De mogelijke uitkomsten ("waar" of "onwaar") van elke conditie en van de beslissing worden minimaal één keer getest
1	1	1	
1	0	1	
0	1	0	
0	0	0	

Als aantal boeken > 8 OF bedrag ≥ 100 DAN Extra korting

	Aantal boeken > 8	Bedrag ≥ 100	Uitkomst
TS1	1	1	1 (extra korting)
TS2	0	0	0 (geen korting)

De uitkomst van de conditie Aantal Boeken > 8 is één keer WAAR en één keer ONWAAR

De uitkomst van beslissing is één keer WAAR en één keer ONWAAR

Voor Condition/Decision coverage zijn er maar twee testsituaties per beslispunt nodig.

Hierbij is het van belang om te variëren in de uitkomst van het beslispunt, evenals in die van de condities. Omdat er slechts twee mogelijke uitkomsten van een beslispunt zijn (DAN of ANDERS) en er ook slechts twee uitkomsten van een conditie zijn (waar of onwaar), kunnen de testsituaties zodanig gedefinieerd worden dat er slechts twee testsituaties per beslispunt nodig zijn.

#### 3.4.2.4 Modified Condition/Decision coverage

Met Modified Condition/Decision coverage (MCDC) is elke mogelijke uitkomst van een conditie minimaal één keer bepalend voor de uitkomst van de beslissing. Dit impliceert "Condition/Decision coverage". Met andere woorden: we zorgen dat voor elke conditie dat als deze de uitkomst "waar" heeft dat dan resulteert in een uitkomst "waar" voor het gehele beslispunt, en dat als deze conditie de uitkomst "onwaar" heeft dat dan resulteert in een uitkomst "onwaar" voor het gehele beslispunt.

MCDC garandeert:

- 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;
- dat ditzelfde geldt voor alle andere condities in het beslispunt.

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. Vergelijken 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.

#### Neutrale waarde voor EN

A	EN	B	Uitkomst
1	.	.	1
0	.	.	0

Laten we zeggen dat A de bepalende conditie is. A kan WAAR en ONWAAR zijn. Dus als A WAAR is, moet de uitkomst ook WAAR worden. Als A ONWAAR is, moet de uitkomst ook ONWAAR worden

**Modified Condition/Decision Coverage**  
Elke mogelijke uitkomst van een conditie bepaalt op zijn minst één keer de uitkomst van de beslissing.

Op deze plaatsen moet een waarde worden toegevoegd, die geen invloed heeft op de uitkomst van de beslissing

A	EN	B	Uitkomst
1	.	1	1
0	.	0/1	0

Als A onwaar is, kunnen beide waarden hier worden ingevoegd.  
Omdat de neutrale waarde moet gelden voor allebei de uitkomsten van de bepalende conditie, kiezen we voor waar

Neutrale waarde voor EN is 1

### Neutrale waarde voor OF

A	EN	B	Uitkomst
1	.	.	1
0	.	.	0

Laten we zeggen dat A de bepalende conditie is. A kan WAAR en ONWAAR zijn. Dus als A WAAR is, moet de uitkomst ook WAAR worden. Als A ONWAAR is, moet de uitkomst ook ONWAAR worden

### Modified Condition/Decision Coverage

Elke mogelijke uitkomst van een conditie bepaalt op zijn minst één keer de uitkomst van de beslissing.

A	EN	B	Uitkomst
1	.	0/1	1
0	.	0	0

Op deze plaatsen moet een waarde worden toegevoegd, die geen invloed heeft op de uitkomst van de beslissing

Als A onwaar is, kunnen beide waarden hier worden ingevoegd. Omdat de neutrale waarde moet gelden voor allebei de uitkomsten van de bepalende conditie, kiezen we voor onwaar

Neutrale waarde voor OF is 0

### Twee manieren van noteren

A	OF	B	R
1		0	1
0		0	0
0		1	0
0		0	0

De uitkomst van de beslissing voor deze testsituatie is WAAR

$R = A \text{ OF } B$	1	0
A	1 0	0 0
B	0 1	0 0

Omdat de combinatie "0 0" twee keer voorkomt (twee keer dezelfde testsituatie, kunnen we één keer wegstrepen

### Een complexer voorbeeld

**ALS** (type auto = bestelauto **EN** eerste gebruik > 1 Juli 2006) **OF**  
ondernemer = nee

**DAN Belastingplichtig**

	1	0
R = (A <b>EN</b> B) <b>OF</b> C		
A	<u>1..</u>	<u>0..</u>
B	. <u>1.</u>	. <u>0.</u>
C	.. <u>1</u>	.. <u>0</u>

**Drie rijen voor drie condities**

**De bepalende waarden in een diagnoaal**

### Een complexer voorbeeld

**ALS** (type auto = bestelauto **EN** eerste gebruik > 1 Juli 2006) **OF**  
ondernemer = nee

**DAN Belastingplichtig**

	1	0
R = (A <b>EN</b> B) <b>OF</b> C		
A	<u>110</u>	<u>010</u>
B	. <u>1.</u>	. <u>0.</u>
C	.. <u>1</u>	.. <u>0</u>

**De bepalende waarde A is verbonden met B door de operator EN. De neutrale waarde voor EN is 1 (waar)**

**De combinatie van A EN B is waar en verbonden met C door de operator OF. De neutrale waarde van OF is 0**

**Omdat het neutrale waarden zijn, kunnen we ze hier ook toevoegen**

### 3.4.2.5 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. Omdat er slechts twee mogelijke uitkomsten van een conditie zijn (waar of onwaar), is 2 de basis voor het aantal testsituaties dat kan worden

gedefinieerd. Het maximum aantal testsituaties (de volledige beslistabel) hangt af van het aantal condities:  $2^N$ , waarbij N het aantal condities is.

Het met enen en nullen vullen van een tabel kan op vele manieren. Laten we een eenvoudig voorbeeld nemen met drie condities, wat zou leiden tot  $2^3=8$  testsituaties.

We zouden kunnen starten met het vullen van de laatste kolom met 4 keer 01.

Voor de tweede kolom verdubbelen we de 0'n en 1'n (dus 00110011).

Voor de eerste kolom verdubbelen we wederom (dus 00001111).

<b>A</b>	<b>B</b>	<b>C</b>
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Een andere handige manier om de beslissingstabel in te vullen is door gebruik te maken van de zogenaamde "Gray-code". Dit resulteert er in dat er slechts 1 conditie in waarde wijzigt per kolom.

We starten nu met de eerste kolom. Wetende dat we op 8 testsituaties zullen uitkomen, delen we de eerste kolom op in vier 0'n en vier 1'n.

Voor de tweede kolom splitsen we het aantal 0'n en 1'n. Maar daar waar het mogelijk is om de volgorde te "spiegelen" doen we dat. Dus in dit geval na 0011 gaan we door met 1100.

Bij de laatste kolom splitsen we wederom het aantal 0'n en 1'n en ook hier "spiegelen" we weer waar mogelijk. Dus na 01 gaan we door met 10, en daarna weer gespiegeld naar 01, etc.

<b>A</b>	<b>B</b>	<b>C</b>
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

Je ziet nu dat in de tweede regel alleen de waarde C is veranderd ten opzichte van de eerste regel. In de derde regel is alleen de waarde B veranderd ten opzichte van de tweede regel, etc. Dit is handig bij de creatie van de fysieke testgevallen: copy-and-paste en verander slechts 1 waarde.

De Multiple Condition coverage (MCC) kan op twee manieren worden toegepast:

1. Alle combinaties van 0'n en 1'n van condities per beslispunt.
2. Alle combinaties van 0'n en 1'n van alle condities van alle beslispunten.

De testbasis bestaat uit beslistabellen, pseudocode, een procesbeschrijving of andere (functionele) beschrijvingen waarin condities voorkomen. De condities en de resultaten worden dan in een beslistabel gezet:

- 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.

### 3.4.3. Semantiek

Zie paragraaf 3.7.5 (Semantische Test).

## 3.5 Dekkingsvormen Gegevens

### 3.5.1. Inleiding

Gegevens worden gecreëerd en eindigen wanneer ze worden verwijderd. Tussendoor worden de gegevens gebruikt door ze bij te werken of hen te raadplegen. Deze gegevenslevenscyclus kan worden getest, evenals combinaties van invoergegevens en de rubrieken van gegevensinvoer of -uitvoer.

Deze groep bestaat uit de volgende dekkingsvormen:

- Equivalentieklassen (paragraaf 3.5.2)
- Grenswaardenanalyse (paragraaf 3.5.3)
- Gegevenscombinaties (paragraaf 3.5.4)
- Syntax (paragraaf 3.5.5)
- CRUD (paragraaf 3.5.6)
- Integriteitsregels (paragraaf 3.5.7)
- Gegevensstromen (wordt niet expliciet in dit Workbook uitgelegd)
- Domain Testing (wordt niet expliciet in dit Workbook uitgelegd)
- Goedpaden / Foutpaden (wordt niet expliciet in dit Workbook uitgelegd)

### 3.5.2. Equivalentieklassen

#### Kenmerken

Approach	Dekkingsgebaseerd – Gegevens
Kwaliteitsattribuut / Testvariëteit	<ul style="list-style-type: none"><li>• Primair: functionaliteit</li><li>• Maar ook toepasbaar bij andere kwaliteitsattributen of testvariëteiten</li></ul>
Testbasis	<ul style="list-style-type: none"><li>• Vrijwel iedere</li></ul>

#### Beschrijving

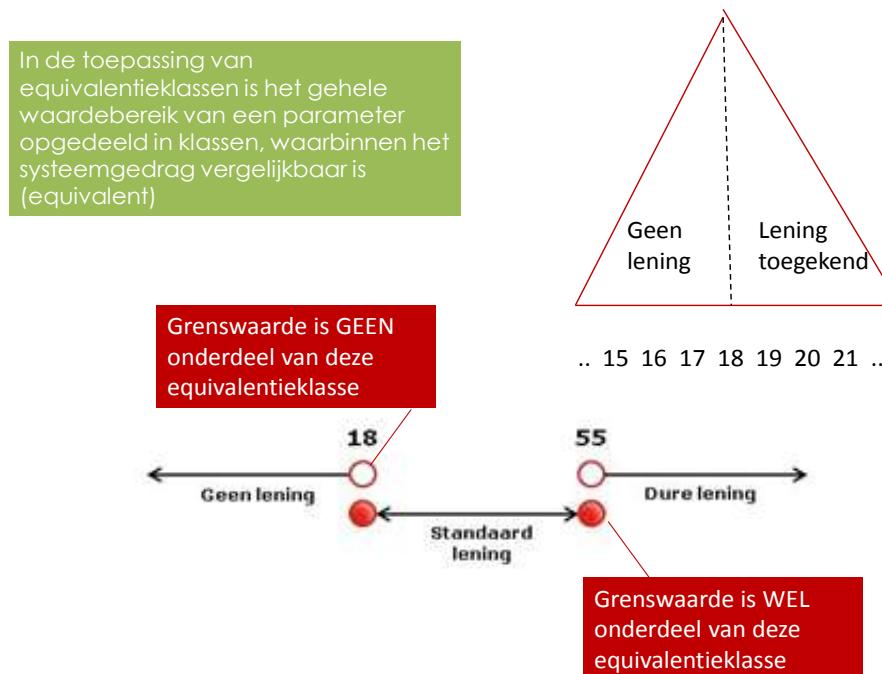
Het dekken van equivalentieklassen is een krachtig middel om met een beperkte set testsituaties een relatief hoge foutvindkans te bereiken. Het principe is eenvoudig en wordt door de meeste ervaren testers vanzelfsprekend en intuïtief toegepast.

Definitie Equivalentieklassen
----------------------------------

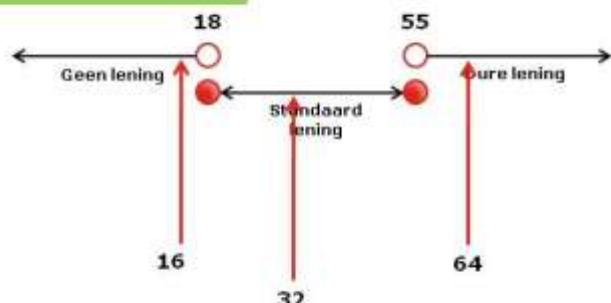
Bij het toepassen van equivalentieklassen wordt het volledige waardebereik van een parameter opgedeeld (gepartitioneerd) in klassen waarbij het systeemgedrag gelijksoortig (equivalent) is.

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 equivalentieklaas een correct systeemgedrag optreedt, is het in principe nog steeds mogelijk dat er een fout optreedt bij een andere waarde.

Ook al is het achterliggende principe een aanname, het is een bruikbare en nuttige aanname. Door test gevallen te baseren op deze equivalentieklassen in plaats van op elke mogelijke invoerwaarde, blijft het aantal test gevallen beperkt, terwijl toch een goede dekking verkregen wordt van de mogelijke variaties in het systeemgedrag.



**Aanname:** elke waarde van een klasse heeft dezelfde kans om een fout te vinden en dat testen met meerdere waarden van dezelfde klasse nauwelijks de kans vergroot om fouten te vinden.



### Mogelijke waarden om te kiezen van de equivalentieklassen

#### 3.5.3. Grenswaardenanalyse

##### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Gegevens
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>Primair: functionaliteit</li> <li>Maar ook toepasbaar bij andere kwaliteitsattributen of testvariëteiten</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>Vrijwel iedere</li> </ul>

##### Beschrijving

Definitie:  
Grenswaarde

Als het systeemgedrag verandert zodra de waarde van een parameter een bepaalde grens overschrijdt, is er sprake van een zogenaamde grenswaarde.

In de praktijk blijken veel fouten te maken te hebben met grenswaarden. Meestal zijn het gewoon 'slordige programmeurfouten' waarbij de programmeur bijvoorbeeld per ongeluk een ">" in plaats van een " $\geq$ " programmeerde, of een "=" in plaats van een " $\geq$ ".

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 equivalentieklas (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 equivalentieklas of conditie;
- definieer de volgende **drie testsituaties**: precies op de grens, direct erboven, direct eronder.

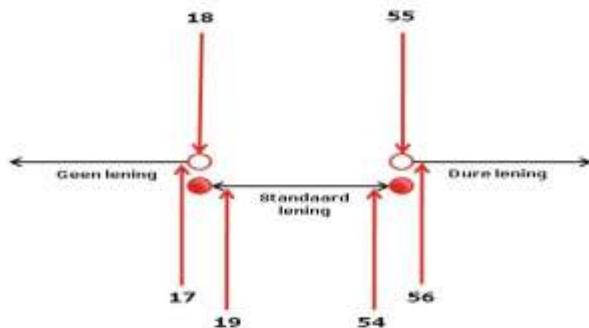
Deze manier van uitwerken van 3 test situaties per grenswaarde heet de normale variant. Er is ook een lichte variant van grenswaardenanalyse, waarbij slechts twee testsituaties getest worden: de grenswaarde zelf plus de daaraan grenzende waarde in de andere equivalentieklas.

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 " $\geq$ " geprogrammeerd is.

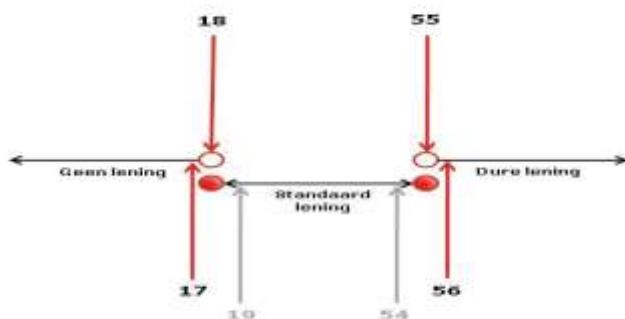
Grenswaardenanalyse is *niet altijd toepasbaar* op equivalentieklassen of condities. Er hoeft namelijk niet altijd sprake te zijn van grenswaarden. Neem bijvoorbeeld de parameter "geslacht" met als waarden (en dus equivalentieklassen) "M" en "V". Er is niet zoziets als een grens tussen de "M" en de "V". Dit geldt bijvoorbeeld ook voor al die parameters in een systeem die behoren tot 'codes' en 'types'.

Hierna volgt stap voor stap een voorbeeld van de toepassing van grenswaardenanalyse. Dit is voor het geval waarbij een persoon een lening kan krijgen als deze de leeftijd van 18 jaar of ouder heeft maar een duurdere lening krijgt zodra deze ouder is dan 55.



#### Normale variant:

- Precies op de grens
- direct erboven
- direct eronder



#### Lichte variant:

- Precies op de grens
- De bijbehorende waarde in de andere equivalentieklassen

### 3.5.4. Gegevenscombinaties

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Gegevens
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Primair: functionaliteit</li> <li>• Maar ook toepasbaar bij andere kwaliteitsattributen of testvariëteiten</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• Vrijwel iedere</li> </ul>

#### Beschrijving

Binnen functionaliteiten kunnen enkelvoudige gegevensattributen (en hun equivalentieklassen) het systeemgedrag beïnvloeden, maar vaak zijn het juist de combinaties van gegevenskenmerken die van invloed zijn op de variaties in het gedrag van het systeem.

Afhankelijk van de afgesproken zwaarte van de test kunnen verschillende varianten gekozen worden voor het testen van dergelijke combinaties:

- Eén of meerdere "data pairs" (het testen van de door de deskundigen aangegeven, bijv. op basis van risico-inschatting, meest interessante paren van gegevens)
- Pairwise testing
- N-wise testing (uitbreiding op Pairwise)
- Alle mogelijke combinaties (= multiple condition coverage toegepast – zie paragraaf 3.4.2 – op gegevens, in plaats van op condities)

### 3.5.4.1 N-Wise en Pairwise

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.

Voor het toepassen van N-wise testing zijn tools noodzakelijk.

#### Pairwise testing

De in de praktijk meest gebruikte vorm van N-wise testing is 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.

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.

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  $2 \times 2 \times 2 = 8$  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 test gevallen, 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 test gevallen. Dat is de kracht van Pairwise testing.

### 3.5.5. Syntax

Zie paragraaf 3.7.4 (Syntactische Test).

### 3.5.6. CRUD

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Gegevens
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>Overkoepelende functionaliteit</li> <li>Inpasbaarheid</li> <li>Connectiviteit</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>CRUD-matrix,</li> <li>Functionele specificaties en/of</li> <li>Gedetailleerde materiedeskundigheid</li> </ul>

#### Beschrijving

Het dekken van alle basisbewerkingen (Create, Read, Update, Delete) op alle entiteiten.

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 onderstaande figuur.

	<b>Factuur</b>	<b>Artikel</b>	...
Beheren artikelen	-	C, U, D	...
Aanmaken factuur	C	R	...
Balie-betaling	C, R, D	-	...
...	...	...	...

### **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;
- welke acties (C, R, U en/of D) op deze entiteiten worden uitgevoerd.

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 specifieke 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:

Verwerkingsfuncties 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 (CRUD-matrix) 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 toets, 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 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)
<b>Deellevering (U)</b>	<b>(veroorzaakt fout in "Voorraadplanning")</b>
Overzicht orders	(R)
<b>Voorraadplanning</b>	<b>(R) (fout wordt ontdekt)</b>
Annuleren order	(D)
Overzicht orders	(R)
Voorraadplanning	(R)

### 3.5.7. Integriteitsregels

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Gegevens
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>Overkoepelende functionaliteit</li> <li>Inpasbaarheid</li> <li>Connectiviteit</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>Beschrijving integriteitsregels,</li> <li>Functionele specificaties en/of</li> <li>Gedetailleerde materiedeskundigheid</li> </ul>

#### Beschrijving

*Integriteitregels beschrijven de voorwaarden onder welke bepaalde CRUD processen wel of niet zijn toegestaan.*

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 dekking van Integriteitregels heeft een sterke relatie met de dekkingsvorm CRUD (Create, Read, Update, Delete). Ze kunnen heel goed samen worden toegepast. Aangezien de integriteitregels kunnen worden geformuleerd als beslispunten, kan ook Decision coverage worden toegepast.

De volgende activiteiten dienen uitgevoerd te worden:

- Verzamel de integriteitregels over de geselecteerde gegevens.  
Dit zijn de regels die definiëren onder welke voorwaarden de bewerkingen op de gegevens geldig zijn of niet. Integriteitregels zijn meestal gespecificeerd in de functionele specificaties, database schema's of in separate 'business rules'.
- Pas Decision Coverage (zie paragraaf 3.4.2) toe. Dat wil zeggen dat voor iedere integriteitregel twee testsituaties afgeleid worden:
  - Ongeldig: Er wordt niet voldaan aan de integriteitregel. De bewerking is ongeldig en moet resulteren in een correcte foutafhandeling.
  - Geldig: Er wordt wel voldaan aan de integriteitregel. De bewerking is geldig en dient uitgevoerd te worden.

#### Voorbeeld

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 "integriteitregel".

## 3.6 Dekkingsvormen Voorkomen

### 3.6.1. Inleiding

De manier waarop een systeem functioneert, hoe het presteert, wat de presentie moet zijn, wordt vaak beschreven in termen van niet-functionele requirements.

Deze groep bestaat uit de volgende dekkingsvormen:

- Presentatie (paragraaf 3.6.2)
- Load profiles (paragraaf 3.6.3)
- Operational profiles (paragraaf 3.6.4)
- Heuristiek (paragraaf 3.6.5)

Andere dekkingsvormen in deze groep (niet dit Workbook uitgewerkt) zijn:

- Gebaseerd op Informatiebeveiliging:
  - Autorisatie
  - Authenticatie
  - Communicatie beveiliging
  - Data confidentialiteit
  - Data integriteit
  - Onweerlegbaarheid
  - Privacy
- Gebaseerd op Bruikbaarheid:
  - Alpha-testing
  - Beta-testing
  - Usability lab

### 3.6.2. Presentatie

Zie paragraaf 3.7.4 (Syntactische Test).

### 3.6.3. Load Profiles

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Voorkomen
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Performance</li> <li>• Continuïteit</li> <li>• Connectiviteit</li> <li>• Bruikbaarheid</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• Beschrijving van het realistisch gebruik (profiles)</li> </ul>

#### Beschrijving

Load profiles 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?"

Load profiles worden vaak toegepast in combinatie met dekkingsvorm Operational 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.

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 vragen: "Welke transacties voer je uit? Hoe vaak en wanneer?"

Het testen van load profiles valt in de categorie "performance testen" 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 overbeladen. 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 zielijk 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 responsijd gemeten wordt. Dit kan in een grafiek uitgezet worden. Op het moment dat de grafiek een scherpe knik omhoog laat zien, neemt de responsijd onevenredig snel toe (de respons 'stort in') en is het breekpunt bereikt.

### 3.6.4. Operational Profiles

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Voorkomen
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>• Performance</li><li>• Continuïteit</li><li>• Connectiviteit</li><li>• Bruikbaarheid</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>• Beschrijving van het realistisch gebruik (profiles)</li></ul>

#### Beschrijving

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 zijn die een significant verschillend statistisch gemiddeld gedrag vertonen, is het verstandig om per type gebruiker een aparte operational profile op te stellen.

Operational profiles worden vaak toegepast in combinatie met dekkingsvorm Load profiles.

### 3.6.5. Heuristiek

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Voorkomen
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>• Gebruikersvriendelijkheid</li><li>• Inpasbaarheid</li><li>• Bruikbaarheid</li><li>• Usability</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>• Beschrijving van het realistisch gebruik (profiles)</li></ul>

#### Beschrijving

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 "heuristiek" 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

## **3.7 Een basisset testontwerptechnieken**

### **3.7.1. Inleiding**

Voor de basisprincipes rondom testontwerptechnieken wordt verwezen naar paragraaf 3.2.2 (De generieke testontwerpstappen) en paragraaf 3.2.5 (Testontwerptechnieken). In deze paragraaf wordt een aantal testontwerptechnieken nader toegelicht:

- Datacombinatietest (paragraaf 3.7.2)
- Procescyclustest (paragraaf 3.7.3)
- Syntactische Test (paragraaf 3.7.4)
- Semantische Test (paragraaf 3.7.5)
- Beslistabeltest (paragraaf 3.7.6)
- Elementaire Vergelijkingentest (paragraaf 3.7.7)
- Gegevenscyclustest (paragraaf 3.7.8)
- Real Life Test (paragraaf 3.7.9)

### **3.7.2. Datacombinatietest (DCT)**

#### **Kenmerken**

<b>Approach</b>	Dekkingsgebaseerd – Gegevens Maar ook: Ervaringsgebaseerd
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>• Overkoepelende functionaliteit</li><li>• Detail-functionaliteit</li></ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"><li>• Equivalentieklassen</li><li>• Eventueel: gegevenscombinaties</li><li>• Eventueel: grenswaardenanalyse</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>• Elke vorm van informatie over de functionaliteit van het systeem<ul style="list-style-type: none"><li>• Waaronder: materiedeskundigheid</li></ul></li></ul>

#### **Beschrijving**

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 standaard hierbij gebruikte dekkingsvorm 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 wordt dekkingsvorm **gegevenscombinaties** gebruikt:

- Eén of meerdere “data pairs” (het testen van de door de deskundigen aangegeven, bijv. op basis van risico-inschatting, meest interessante paren van gegevens)
- Pairwise testing
- N-wise testing (uitbreiding op Pairwise)
- Alle mogelijke combinaties (= multiple condition coverage toegepast – zie paragraaf 3.4.2 – op gegevens, in plaats van op condities)

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 equivalentieklas 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 paragraaf 3.2.2) 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 3.5.2.

- 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 equivalentieklas 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 zichtbaar 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 equivalentiekasse 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.

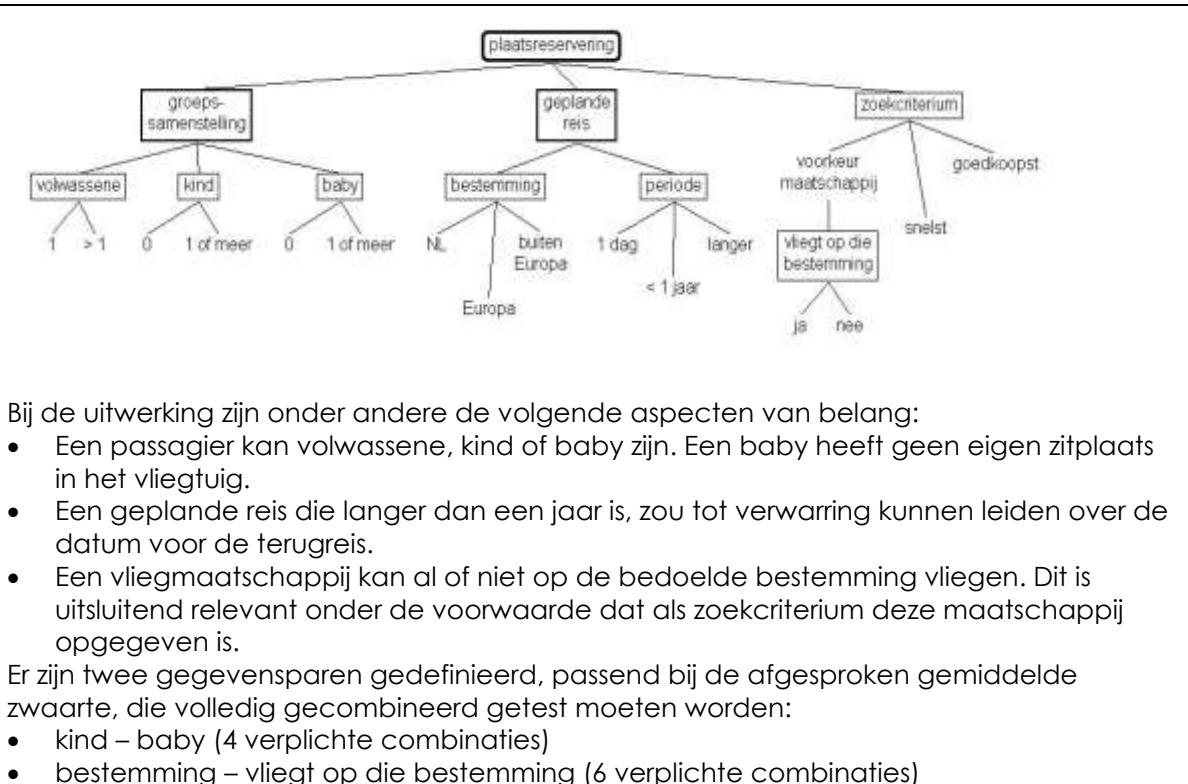
Gemiddeld Twee of meer gegevensparen. Dit biedt een schaal van toenemende zwaarte die eindigt bij "pairwise testing".

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 zwaarte-categorie.

#### Voorbeeld uitwerking

Voor de functie "plaatsreservering" is door het team de volgende classificatieboom uitgewerkt (zie onderstaande figuur):



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:

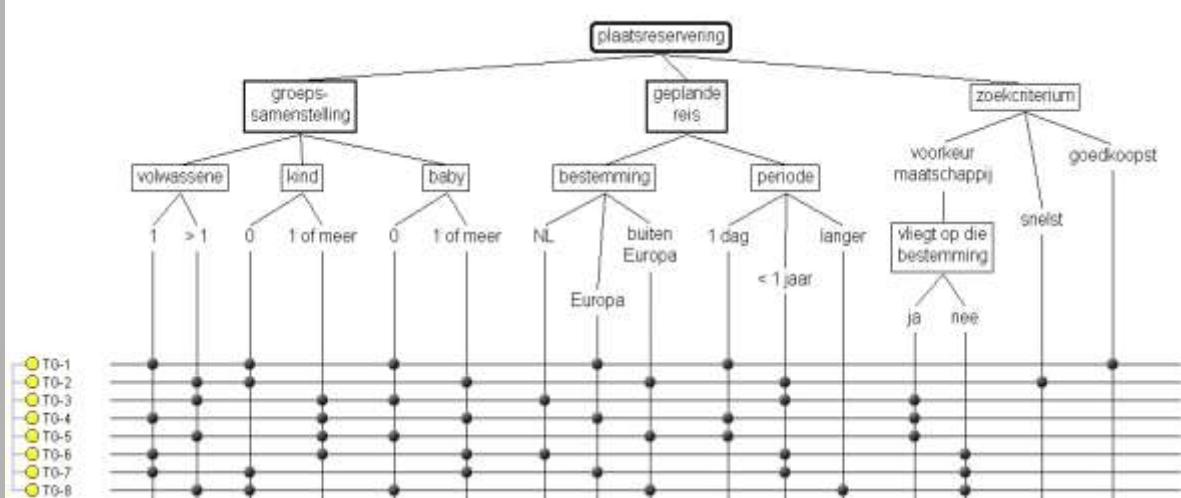
- kind – baby (4 verplichte combinaties)
- bestemming – vliegt op die bestemming (6 verplichte combinaties)

## 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 3.5.4). 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 “Testona” te downloaden. <http://www.berner-mattner.com/de/produkte/testona/index.html>

De logische testgevallen zijn weergegeven met behulp van de classificatieboom (zie figuur):



Hierbij is rekening gehouden met de twee gegevensparen die volledig gecombineerd getest moeten worden. Test gevallen TG-1 t/m TG-4 dekken het gegevenspaar "kind - baby" af, terwijl het gegevenspaar "bestemming - vliegt op bestemming" afgedekt wordt door de test gevallen TG-3 t/m TG-8. De volgende stappen zijn er op zeker te stellen dat elke equivalentieklasse tenminste eenmaal is geraakt en om elke test case volledig te maken zodat voor elke test case elk relevant data attribuut een waarde heeft.

N.B. Voor het behalen van de minimale dekking, waarbij uitsluitend alle equivalentieklassen gedekt worden zonder combinaties van gegevensparen, zijn 4 logische test gevallen hier voldoende, bijvoorbeeld TG-1, TG-2, TG-3 en TG-8.

### 3- Opstellen fysieke test gevallen

Bij het fysiek maken van de test gevallen 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 onderstaande tabel vier test gevallen 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
<b>Resultaatvoorspelling</b>				Melding: "maatschappij vliegt niet op keuzebestemming"
maatschappij	Korean Air	Canada Air	KLM	
vluchtnummer	KA0455	CA0833	KL1288	
prijs	€ 44	€ 865	€ 83	

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.

### 3.7.3. Procescyclustest (PCT)

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Proces
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>Inpasbaarheid</li> <li>Functionaliteit</li> </ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"> <li>Paden testmaat-2</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>AO-beschrijving           <ul style="list-style-type: none"> <li>processschema's</li> <li>beschrijving bedrijfsprocessen / werkprocessen (zoals ook workflows)</li> </ul> </li> <li>Functionele specificaties</li> </ul>

#### Beschrijving

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 variatie van dekkingsvormen die hierbij wordt gebruikt, is:

- Paden: testmaat-2

Variaties op de procescyclustest kunnen worden gecreëerd door het toepassen van de variatie van dekkingsvormen:

- Paden: testmaat-1, testmaat-3 en hoger  
Hiermee kunnen paden respectievelijk lichter of zwaarder worden getest.

### 3.7.3.1 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.

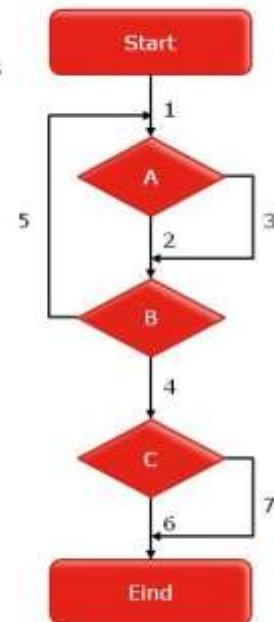
#### 1- Testsituaties identificeren

De eerste stap is het identificeren van padcombinaties op basis van de dekkingsvorm "Paden" (testmaat -2)

A: IN: 1,5  
UIT: 2,3  
  
B: IN: 2,3  
UIT: 4,5  
  
C: IN: 4  
UIT: 6, 7

Pad combinaties- level 2				
A	1-2	1-3	5-2	5-3
B	2-4	2-5	3-4	3-5
C	4-6	4-7		

Laten we doorgaan met ons voorbeeld van paden.  
Deze padcombinaties zijn onze testsituaties.



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

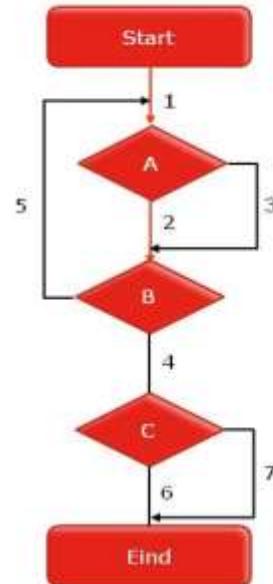
het schema de testsituaties afgeleid volgens de beschreven techniek bij de dekkingsvorm Paden.

### 3.7.3.2 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". De tester is vrij om te kiezen op welke manier het proces wordt doorlopen, mits alle testsituaties minimaal 1x worden afgedekt.



Indien noodzakelijk kan met behulp van een cross-reference matrix worden aangetoond dat alle testsituaties zijn gedekt met de gekozen set van testgevallen. In principe zijn er 2 manieren om te komen tot een dekkende set van testgevallen:

- Uitgaande van het proces schema, definieer een testgeval door het proces op een bepaalde manier van "Start" tot "Eind" te doorlopen. De tester is hierbij vrij om te kiezen op welke manier het proces wordt doorlopen. Streep de pad combinaties af die in dit testgeval zijn gedekt. Herhaal dit totdat alle pad combinaties zijn afgestreept.
- Uitgaande van de lijst met pad combinaties, start met een pad combinatie die begint bij "Start". Zoek een volgende pad combinatie die start met het pad waarmee de voorgaande pad combinatie eindigde – zoets als het leggen van domino stenen. Ga door met zoeken naar volgende pad combinaties totdat "Eind" is bereikt. Hierbij dienen natuurlijk zoveel mogelijk de nog niet eerder gebruikte padcombinaties worden gebruikt.

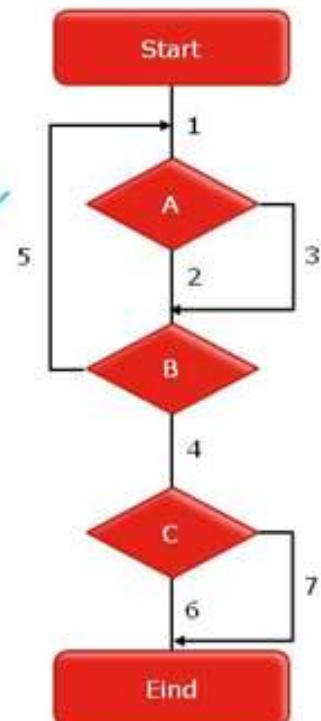
## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen

Ga door het proces van "start" naar "eind", zo lang alle testsituaties tenminste eens worden afgedekt

A:	IN	1, 5
	UIT	2, 3
B:	IN	2, 3
	UIT	4, 5
C:	IN	4
	UIT	6, 7

Padcombinaties – testmaat 2					
	A	1-2	1-3	5-2	5-3
A	2	1-2	1-3	5-2	5-3
B	4	2-4	2-5	3-4	3-5
C	6	4-6	4-7		

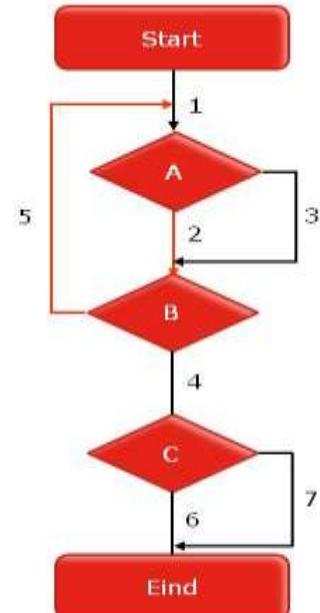


## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen

A:	IN	1, 5
	UIT	2, 3
B:	IN	2, 3
	UIT	4, 5
C:	IN	4
	UIT	6, 7

Padcombinaties – testmaat 2					
	A	1-2	1-3	5-2	5-3
A	2	1-2	1-3	5-2	5-3
B	4	2-4	2-5	3-4	3-5
C	6	4-6	4-7		

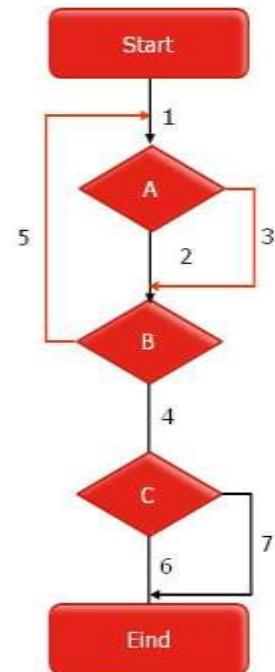


## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen

A: IN 1, 5  
UIT 2, 3  
IN 2, 3  
UIT 4, 5  
IN 4  
UIT 6, 7

Padcombinaties – testmaat 2					
	A	1-2	1-3	5-2	5-3
B	2-4	2-5	3-4	3-5	
C	4-6	4-7			

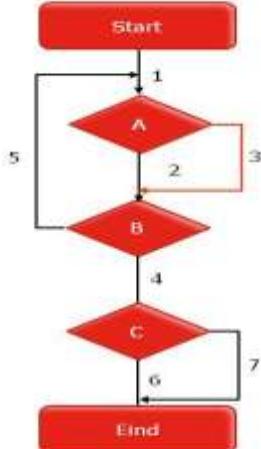


## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen

A: IN 1, 5  
UIT 2, 3  
IN 2, 3  
UIT 4, 5  
IN 4  
UIT 6, 7

Padcombinaties – testmaat 2					
	A	1-2	1-3	5-2	5-3
B	2-4	2-5	3-4	3-5	
C	4-6	4-7			

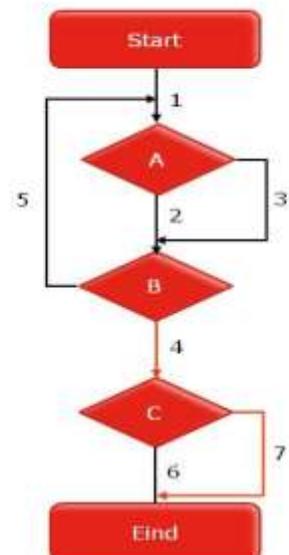


## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen

A: IN 1, 5  
UIT 2, 3  
  
B: IN 2, 3  
UIT 4, 5  
  
C: IN 4  
UIT 6, 7

Padcombinaties – testmaat 2				
	A	1-2	1-3	5-2
B	2-4	2-5	3-4	3-5
C	4-6	4-7		



## 2. Logische testgevallen maken

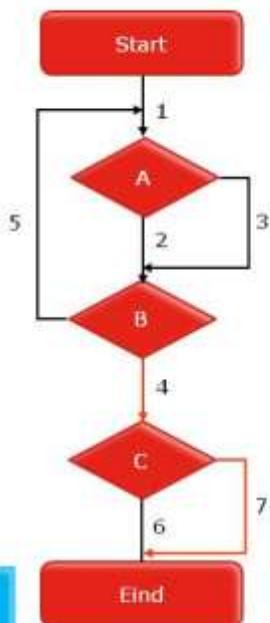
- Creëren van een set logische testgevallen

A: IN 1, 5  
UIT 2, 3  
  
B: IN 2, 3  
UIT 4, 5  
  
C: IN 4  
UIT 6, 7

Padcombinaties – testmaat 2				
	A	1-2	1-3	5-2
B	2-4	2-5	3-4	3-5
C	4-6	4-7		

Logische Testgevallen	
TC1	1-2-5-3-4-7

Hou dit vol totdat alle testsituaties (padcombinaties) zijn afgedekt



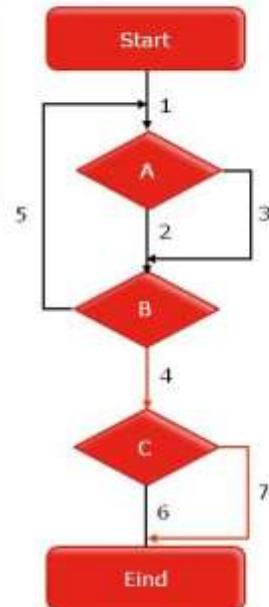
## 2. Logische testgevallen maken

- Creëren van een set logische testgevallen
- Beschrijven van de vervolgacties per logische testcase

Logische Testgevallen	
TC1	1-2-5-3-4-7
TC2	1-3-5-2-4-6

TC (1-2-5-3-4-7)

- |      |   |              |
|------|---|--------------|
| A1-1 | Aanmaken claim voor   | (Verzekerde) |
| A1-2 | Invoegen van claim formulier details in het systeem<br>(incompleet)   | Werknemer    |
| A1-3 | Start het proces 'check op volledigheid'                              | Werknemer    |
| A1-4 | Neem contact op met de verzekerde partij om de details te completeren | Werknemer    |
| Etc. |   |              |



### 3.7.3.3 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.

### 3. Fysieke testgevallen maken

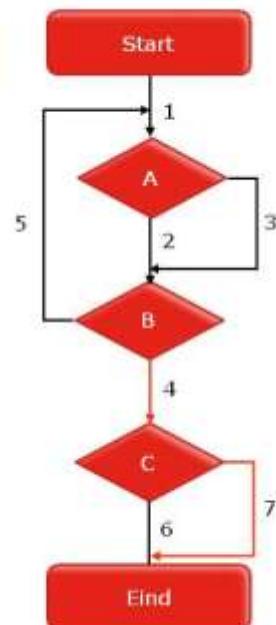
TC (1-2-5-3-4-7)

#### Verzekerde partij

- A1-1 Invullen claim formulier met de volgende details:  
-Naam: Smit, J.  
-Etc.

#### Werknemer

- A1-2 Voeg claim formulier details in in het systeem (zonder 'datum verlies' en zonder 'omschrijving van verlies')  
A1-3 Start het proces 'check op volledigheid' (formulier is incompleet)  
A1-4 Neem contact op met de verzekerde partij om 'datum verlies' te verkrijgen.  
Etc.



#### 3.7.3.4 Bepalen uitgangssituatie

Geen bijzonderheden.

#### 3.7.3.5 Maken van het testscript

Over het algemeen is het zo dat het opstellen van het testscript eenvoudig is omdat een fysiek testgeval al een volledig scenario omvat.

### 3.7.4. Syntactische Test (SYN)

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd – Gegevens Dekkingsgebaseerd – Voorkomen
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>• Functionaliteit (Validatietest)</li><li>• Gebruikersvriendelijkheid (Presentatietest)</li></ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"><li>• Syntax</li><li>• Presentatie</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>• Data dictionary of andere gegevensmodellen</li><li>• Stijlgidsen</li><li>• Lijst- en schermspecificaties</li></ul>

#### Beschrijving

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.

Daarnaast is de Syntactische test ook een *presentatietest*. Deze test richt zich op opmaakcontroles en overige lay-outaspecten.

#### Validatietest

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. Bij deze validatietests wordt dekkingsvorm Syntax toegepast.

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.

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.

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, inclusief de waarde 0, maar exclusief 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).

### **Presentatietest**

Presentatietests zijn gericht op het beoordelen van de layout van bijvoorbeeld schermen en overzichten. Hierbij wordt dekkingsvorm Presentatie toegepast.

Overzicht van layoutcontroles:

- Kopregels/Voetregels
  - Wordt er aan de standaards voldaan?  
Bijv. standaards voor scherm- of lijstnaam, systeem- of afdrukdatum, versienummer
- Rubrieken
  - Per rubriek zijn er vaak specifieke eisen aan de opmaak gedefinieerd, bijvoorbeeld:  
Naamgeving van de rubriek, plaats van de rubriek op scherm of overzicht (bijv. plaats van adres als de brief in een vensterenvelop gaat) of de weergave van de rubriek (zoals lettertype, kleur, etc.)

### **3.7.5. Semantische Test (SEM)**

#### **Kenmerken**

<b>Approach</b>	Dekkingsgebaseerd – Condities
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Functionaliteit (Validatietest)</li> </ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"> <li>• Semantiek =&gt; Beslispunten: Modified Condition / Decision Coverage</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• Functionele specificaties</li> <li>• Overkoepelende 'business rules'</li> </ul>

#### **Beschrijving**

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 3.7.4).

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 paragraaf 3.2.2) 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 ongeldige 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 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 resultert 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 waarheidstabellen voor deze operator – die weergegeven wordt met het symbool “ $\rightarrow$ ” - is:

A	B	$A \rightarrow B$
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 waarheidstabellen:

“ $A \rightarrow B$ ” 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 dekkingsvorm 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  $\geq 55$ ”

Hierin is een imply-operator toegepast, waardoor de regel er feitelijk uitziet als:

“code\_bijdrage = V  $\rightarrow$  (code\_dienstverband = F EN leeftijd  $\geq 55$ )”

Deze kan omgezet worden naar de volgende samengestelde conditie:

“(NOT code\_bijdrage = V) OF (code\_dienstverband = F EN leeftijd  $\geq 55$ )”

oftewel

“code\_bijdrage ≠ V OF (code\_dienstverband = F EN leeftijd  $\geq 55$ )”

Toepassen van dekkingsvorm 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 $\geq 55$	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.

### 3.7.6. Beslistabeltest (BTT)

#### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd - condities (eventueel: data)
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>Functionaliteit<ul style="list-style-type: none"><li>Detailfunctionaliteit (zeer belangrijk geachte functies en/of complexe berekeningen)</li><li>Grondig afdekken van de condities</li><li>Niet: het combineren van functionele paden</li></ul></li></ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"><li>Beslispunten: Multiple Condition Coverage</li><li>Eventueel: Gegevenscombinaties</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>Beslistabellen, pseudo-code, een procesbeschrijving of andere (functionele) beschrijvingen, waarin condities voorkomen</li></ul>

#### Beschrijving

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 beslistabeltest. De beslistabeltest richt zich op het grondig afdekken van de condities en niet op het combineren van functionele paden. De dekkingsvorm die hierbij wordt gebruikt, is:

- Beslispunten: multiple condition coverage

Variaties op de beslistabeltest kunnen worden gecreëerd door het toepassen van andere dekkingsvormen:

- 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

De beslistabeltest wordt hier stap voor stap toegelicht. Hierbij zijn de generieke stappen (zie paragraaf 3.2.2. De generieke [testontwerpstappen](#)) het uitgangspunt. 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. Hieronder is de algemene vorm van een beslistabel weergegeven.

Identificatie tabel				
Testsituaties	1	2	..	$2^n$
Conditie 1	0	0	..	1
Conditie 2	0	~	..	0
Conditie ..	0	..	~	0
Conditie n	0	1	..	0
Resultaat 1	X	..	..	..
Resultaat ..	..	..	..	..
Resultaat m	..	..	..	..

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 3.4.2). 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.

#### 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 op dit voorbeeld wordt toegepast:

### 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  $\geq 200$  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  $\geq$  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 dekkingsvorm Beslispunten: Multiple Condition Coverage gevuld. Dat wil dus zeggen: alle mogelijke combinaties van waarden (0 of 1) van de afzonderlijke condities.

### Voorbeeld

Aangezien het totaal aantal condities vier bedraagt, heeft de tester besloten deze in één tabel op te nemen<sup>1</sup>. 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 onderstaand tabel:

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 $\geq$ 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 Utr.	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 Afstandtoeslag := 0			X	X	X	X				X	X	X	X			
R4 Afstandtoeslag := 5		X				X			X					X		
R5 Afstandtoeslag := 15	X			X	X			X	X		X	X				X

Het lezen van een tabel wordt vaak lastig gevonden. Testsituatie 7 bijvoorbeeld, moet als volgt worden 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 afstandtoeslag 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

---

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  $2^4 = 16$  combinaties. In het voorbeeld zou de tabel in twee tabellen ("berekening standaardverzendkosten" en "berekening afstandtoeslag") kunnen worden gesplitst. Beide tabellen zouden dan uit  $2^2 = 4$  testkolommen bestaan. Die in combinatie met elkaar  $4 \times 4$ , 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.

In plaats van het vullen van de beslistabel volgens dekkingsvorm Beslispunten: Multiple Condition Coverage, kan in de teststrategie zijn vastgelegd dat er lichter mag worden getest. In dat geval wordt voor een lichtere dekkingsvorm gekozen, bijvoorbeeld Beslispunten: Condition / Decision Coverage. Deze dekkingsvorm wordt op zowel de condities als de resultaten toegepast. Als voorbeeld zijn hieronder de testsituaties uitgewerkt voor "verzendkostenberekening":

Verzendkostenberekening (testsituaties)			
	1	2	11
C1 Bestelling $\geq$ 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. verz.kosten := 0			X
R2 St. Verzendkosten := 10	X	X	
R3 Afstandstoeslag := 0			X
R4 Afstandstoeslag := 5		X	
R5 Afstandstoeslag := 15	X		

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 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). Het testen van alle combinaties valt dan als dekkingsvorm onder de groep Data – Gegevenscombinaties (in plaats van Condities – Beslispunten; of, zoals in onderstaand uitgewerkt voorbeeld, onder beide groepen):

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:

	1	2	3
C1 Bestelling $\geq$ 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

## 2- Opstellen logische testgevallen

De testsituaties (kolommen) uit stap 1 zijn direct de logische testgevallen. Een logisch testgeval kan 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 onderstaande tabel.

**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 $\geq$ 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 Utr.	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 Afstandtoeslag := 0			X		X					X			X			
R4 Afstandtoeslag := 5		X				X			X						X	
R5 Afstandtoeslag := 15	X						X	X								X
Niet mogelijk				X	X							X	X			

Ga zelf na dat de logische testgevallen 4, 5, 12 en 13 niet uitvoerbaar zijn.

De logische testgevallen met als resultaat "Niet mogelijk" vervallen als logisch testgeval. Deze logische testgevallen worden echter in principe *niet* uit de tabel verwijderd, om aan te tonen dat de juiste dekkingsvorm is toegepast. De overblijvende logische testgevallen worden 1-op-1 uitgewerkt in fysieke testgevallen.

**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). Bij het invullen wordt:

- elke "0" of de "1" uit de tabel vervangen door een fysieke waarde;
- in de resultaatvoorspelling op de plaats van elke "X" een 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**

Voor het fysiek maken is het van "Afstand van Utrecht" afgeleide gegeven "Afleverplaats" nodig. Het resultaat is weergegeven in onderstaande tabel, 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	60	178	182	10	178
Afleverplaats	Brussel	Heerlen	Zeist	Arnhem	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

De logische testgevallen 4, 5, 12 en 13 zijn niet uitvoerbaar en kunnen dus ook niet fysiek worden gemaakt.

#### Uitgediept

De beslistabeltest kan worden verwaard door het toepassen van Grenswaardenanalyse. De te testen grenswaarden worden dan als een extra set testsituaties 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 7, 9 en 10) en bij afstand in ieder geval de waarden 49, 50 en 51 (bijvoorbeeld in de kolommen 6, 7 en 2) in de testgevallen voorkomen. Het aantal testgevallen verandert bij deze aanpak niet (de afleverplaats natuurlijk wel).

Een andere mogelijkheid is om voor elke grenswaarde een aparte kolom op te nemen. Dit is een grondige methode die alle combinaties van de grenswaarden en de overige condities test. Dit is zeer arbeidsintensief, waarbij het aantal testgevallen wél toeneemt.

#### 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.

#### 3.7.7. Elementaire Vergelijkingstest (EVT)

##### Kenmerken

<b>Approach</b>	Dekkingsgebaseerd - Condities
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>Functionaliteit <ul style="list-style-type: none"> <li>Detailfunctionaliteit</li> <li>Grondig afdekken van de beslispunten</li> <li>Niet: het combineren van functionele paden</li> </ul> </li> </ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"> <li>Beslispunten: Modified Condition / Decision Coverage</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>Beslistabellen, pseudo-code, een procesbeschrijving of andere (functionele) beschrijvingen, waarin condities van beslispunten voorkomen.</li> </ul>

## **Beschrijving**

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 dekkingsvormen die hierbij gebruikt worden zijn:

- Beslispunten: Modified Condition / Decision coverage

Variaties op de EVT kunnen gecreëerd worden door het toepassen van de volgende dekkingsvormen:

- Beslispunten: Multiple Condition coverage  
Hiermee kunnen de mogelijkheden binnen de (eventueel specifiek geselecteerde) beslispunten zwaarder getest worden.
- Beslispunten: Condition coverage t/m Condition / Decision coverage  
Hiermee kunnen de mogelijkheden binnen de (eventueel specifiek geselecteerde) beslispunten minder zwaar 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.

### **3.7.7.1 Aandachtspunten voor de vijf generieke stappen**

#### **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).

Per beslispunt wordt de dekkingsvorm beslispunten (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" (waar) en daarna die uit de kolom "0" (onwaar).

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.



ALS	leeftijd < 18 jaar <b>OF</b> rijbewijs ingevorderd
DAN	Foutbericht
ANDERS	
<b>ALS</b>	leeftijd < 25 jaar <b>EN</b> aantal jaren rijbewijs < 3
DAN	premie := 1.500
ANDERS	premie := 800
EINDALS	
<b>ALS</b>	autoleeftijd < 2 <b>OF</b> (autoleeftijd ≥ 5 <b>EN</b> schade in afgelopen 3 jaar ≥ 2.500) <b>OF</b> leeftijd ≥ 70
DAN	verhoog premie met 500
EINDALS	
EINDALS	

Dit beslispunt en het volgende zijn 'genest' in het eerste beslispunt.

<b>ALS</b>	leeftijd < 18 jaar <b>OF</b> rijbewijs ingevorderd
DAN	Foutbericht
ANDERS	
<b>ALS</b>	leeftijd < 25 jaar <b>EN</b> aantal jaren rijbewijs < 3
DAN	premie := 1.500
ANDERS	premie := 800
EINDALS	
<b>ALS</b>	autoleeftijd < 2 <b>OF</b> (autoleeftijd ≥ 5 <b>EN</b> schade in afgelopen 3 jaar ≥ 2.500) <b>OF</b> leeftijd ≥ 70
DAN	verhoog premie met 500
EINDALS	
EINDALS	

### **1- testsituaties identificeren**

- Beslispunten identificeren (zoek naar ALS-delen)
- Geef de beslispunten een unieke identificatie

<b>D1</b>	<b>ALS</b>	leeftijd < 18 jaar <b>OF</b> rijbewijs ingevorderd Foutbericht
<b>D2</b>	<b>ANDERS</b>	<b>ALS</b> leeftijd < 25 jaar <b>EN</b> aantal jaren rijbewijs < 3 DAN premie := 1.500 ANDERS premie := 800 EINDALS
<b>D3</b>		<b>ALS</b> autoleeftijd < 2 <b>OF</b> (autoleeftijd ≥ 5 <b>EN</b> schade in afgelopen 3 jaar ≥ 2.500) <b>OF</b> leeftijd ≥ 70 DAN verhoog premie met 500 EINDALS

<b>D1</b>	<b>ALS</b>	leeftijd < 18 jaar <b>OF</b> rijbewijs ingevorderd foutbericht
	<b>DAN</b>	
	<b>ANDERS</b>	

### **1- testsituaties identificeren**

- Beslispunten identificeren (zoek naar ALS-delen)
- Geef de beslispunten een unieke identificatie
- Werk ieder beslispunt in detail uit volgens de dekkingsvorm Modified Condition/Decision Coverage.
- Geef elke testsituatie een unieke identificatie

<b>D1 A OF B</b>	<b>1 foutmelding!</b>	<b>0 (D2)</b>
A: leeftijd < 18	1 0 (1-1)	0 0 (1-3)
B: rijbewijs ingevorderd	0 1 (1-2)	-0-0

Geeft de uitkomst aan (wat ook kan inhouden dat er naar het volgende beslispunt moet worden gegaan)

**D1** ALS  
DAN  
ANDERS leeftijd < 18 jaar **OR** rijbewijs ingevorderd  
foutbericht

<b>D1 A OF B</b>	<b>1 foutboodschap</b>	<b>0 (D2)</b>
A: leeftijd < 18	1 0 (1-1)	0 0 (1-3)
B: rijbewijs ingevorderd	0 1 (1-2)	0 0

D2 ALS Leeftijd < 25 jaar EN aantal jaren rijbewijs < 3  
DAN premie := 1.500  
ANDERS premie := 800

<b>D2 A EN B</b>	<b>1 premie = 1500</b>	<b>0 premie = 800</b>
A: Leeftijd < 25	1 1 (2-1)	0 1 (2-2)
B: aantal jaren rijbewijs < 3	1 1 (2-1)	1 0 (2-3)

**D3** ALS                    autoleeftijd < 2 **OF** (autoleeftijd  $\geq 5$   
**AND** schade in de afgelopen 3 jaar  $\geq 2500$ )  
**OR** leeftijd  $\geq 70$   
DAN                    Verhoog premie met 500

<b>D3 A OR (B AND C) OR D</b>	<b>1 premie + 500</b>	<b>0</b>
A: autoleeftijd < 2	1 0 1 0        (3-1)	0 0 1 0        (3-4)
B: autoleeftijd $\geq 5$	0 1 1 0        (3-2)	0 0 1 0
C: schade in de afgelopen 3 jaar $\geq 2500$	<del>0 1 1 0</del>	<del>0 1 0 0</del> (3-5)
D: leeftijd $\geq 70$	0 1 0 1        (3-3)	<del>0 1 0 0</del>

**D1** ALS                    leeftijd < 18 jaar **OF** rijbewijs ingevorderd  
DAN                    foutbericht  
ANDERS

<b>D1 A OF B</b>	<b>1 foutbericht</b>	<b>0 (D2)</b>
A: leeftijd < 18	1 0        (1-1)	0 0        (1-3)
B: rijbewijs ingevorderd	0 1        (1-2)	<del>0 0</del>

### 1- testsituaties identificeren

- Beslispunten identificeren (zoek naar ALS-delen)
- Geef de beslispunten een unieke identificatie
- Werk ieder beslispunt in detail uit volgens de dekkingsvorm Modified Condition/Decision Coverage.
- Geef elke testsituatie een unieke identificatie
- Gedetailleerde uitwerking van de verkregen testsituaties

**D1** ALS  
DAN  
ANDERS      leeftijd < 18 jaar **OF** rijbewijs ingevorderd  
foutboodschap

<b>D1 A OF B</b>	<b>1 foutboodschap</b>	<b>0 (D2)</b>
A: leeftijd < 18	1 0      (1-1)	0 0      (1-3)
B: rijbewijs ingevorderd	0 1      (1-2)	<del>0 0</del>



<b>D1</b>	<b>D1-1</b>	<b>D1-2</b>	<b>D1-3</b>
Leeftijd	< 18	≥ 18	≥ 18
Rijbewijs ingevorderd	N	Y	N

<b>D1</b>	<b>D1-1</b>	<b>D1-2</b>	<b>D1-3</b>
Leeftijd	< 18	≥ 18	≥ 18
Rijbewijs ingevorderd	N	J	N

<b>D2</b>	<b>D2-1</b>	<b>D2-2</b>	<b>D2-3</b>
Leeftijd	< 25	≥ 25	< 25
#jaren rijbewijs	< 3	< 3	≥ 3

<b>D3</b>	<b>D3-1</b>	<b>D3-2</b>	<b>D3-3</b>	<b>D3-4</b>	<b>D3-5</b>
Autoleeftijd	< 2	≥ 2	≥ 2	≥ 2	≥ 2
Autoleeftijd	< 5	≥ 5	≥ 5	< 5	≥ 5
3-jaarschade	≥ 2.500	≥ 2.500	< 2.500	≥ 2.500	< 2.500
Leeftijd	< 70	< 70	≥ 70	< 70	< 70

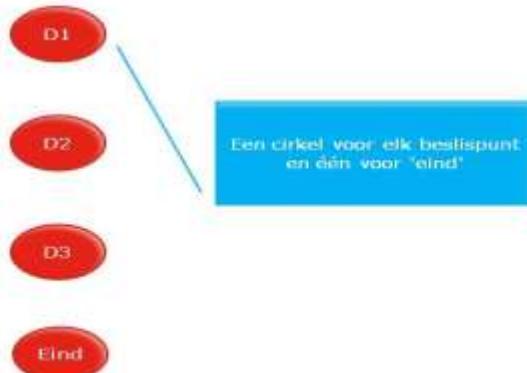
## Grafische voorstelling van testsituaties

Voor sommige testers wordt het bedenken van logische testgevallen eenvoudiger met behulp van een grafische voorstelling van de testsituaties – kortweg: Graaf.

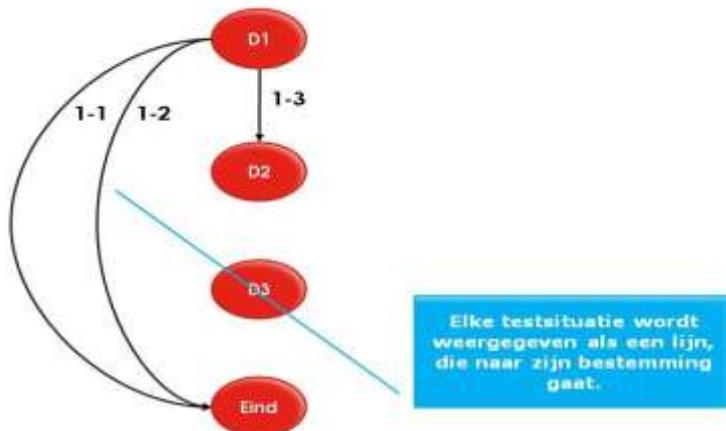
Daarbij wordt ieder beslispunt en eindpunt weergegeven door een cirkel en iedere testsituatie door een lijn die van de ene cirkel naar een andere gaat. Een logisch testgeval doorloopt de graaf van begin tot eind, via een aaneenschakeling van testsituaties. Dit wordt namelijk bepaald door het maximaal aantal parallelle lijnen in de graaf.

### 1- testsituaties identificeren

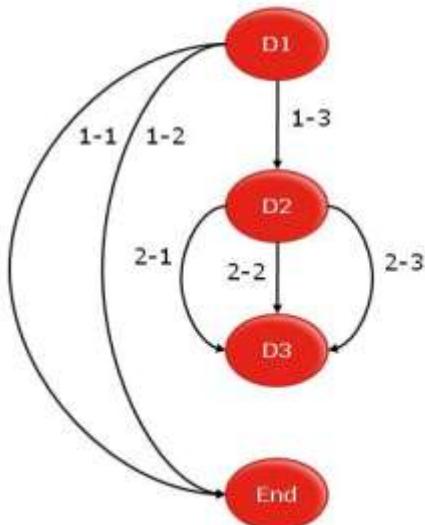
- Beslispunten identificeren (zoek naar AFS-delen)
- Geef de beslispunten een unieke identificatie
- Werk ieder beslispunt in detail uit volgens de dekkingsvorm Modified Condition/Decision Coverage.
- Geef elke testsituatie een unieke identificatie
- Gedetailleerde uitwerking van de verkregen testsituaties
- Grafische weergave van testsituaties



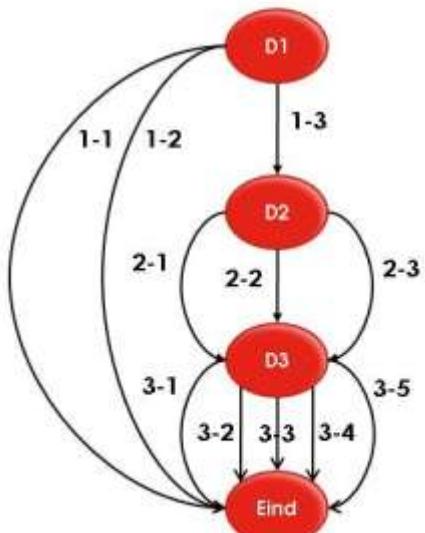
D1 A OF B	1 foutbericht	0 (D2)
A: leeftijd < 18	1 0 (1-1)	0 0 (1-3)
B: rijbewijs ingevorderd	0 1 (1-2)	0 0



D2 A EN B	<b>1</b> Premie = 1500	<b>0</b> Premie = 800
A: leeftijd < 25	1 1 (2-1)	0 1 (2-2)
B: aantal jaren rijbewijs < 3	←→	1 0 (2-3)



D3 A OF (B EN C) OF D	<b>1</b> Premie + 500	<b>0</b>
A: autoleeftijd < 2	1 0 1 0 (3-1)	0 0 1 0 (3-4)
B: autoleeftijd ≥ 5	0 1 1 0 (3-2)	<del>0 0 1 0</del>
C: Schade in de laatste 3 jaar ≥ 2500	<del>0 1 1 0</del>	0 1 0 0 (3-5)
D: leeftijd ≥ 70	0 1 0 1 (3-3)	<del>0 1 0 0</del>



## 2. Opstellen logische testgevallen

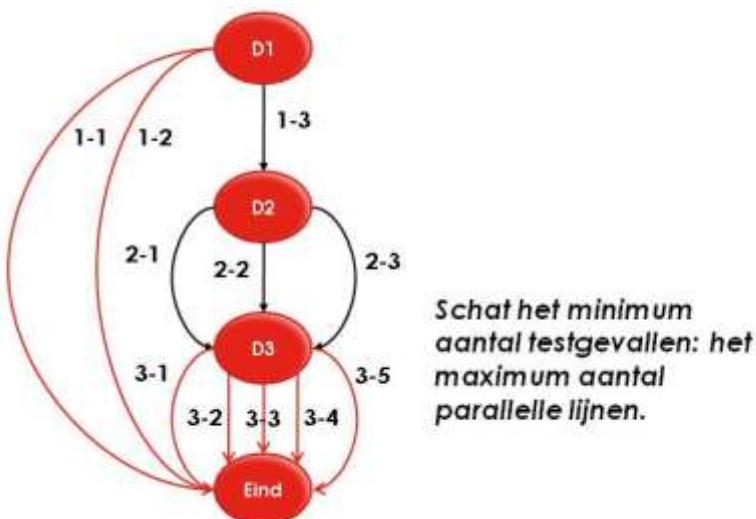
Een testgeval doorloopt de functionaliteit van start tot eind en zal daarbij één of meer beslispunten op zijn pad tegenkomen. Bij ieder beslispunt zal het testgeval één van de gedefinieerde testsituaties testen.

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 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.

### 2- Logische testgevallen maken

- Bepaal testsituaties die elkaar uitsluiten



### Elkaar uitsluitende testsituaties

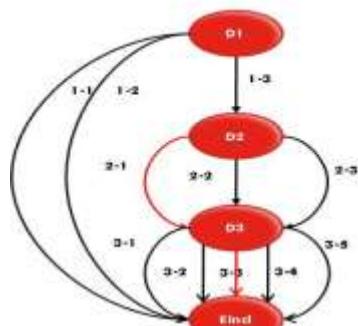
Iedere testsituatie stelt bepaalde eisen aan één of meer parameters. Als een parameter in meerdere beslispunten voorkomt, kan het gebeuren dat een testsituatie uit het ene beslispunt eisen aan die parameter stelt die strijdig zijn met de eisen van een testsituatie uit een ander beslispunt.

Een logisch testgeval mag geen "elkaar uitsluitende testsituaties" bevatten, want dat maakt het testgeval inconsistent en daarmee onuitoefbaar. 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 het ontstaan van testgevallen met elkaar uitsluitende testsituaties te voorkomen, 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.

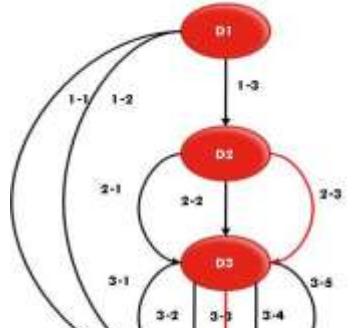


D1	D1-1	D1-2	D1-3
Leeftijd	<18	≥18	≥18
Rijbewijs ingevorderd	N	J	N

D2	D2-1	D2-2	D2-3
Leeftijd	<25	≥25	<25
#jaren rijbewijs	<3	<3	≥3

D3	D3-1	D3-2	D3-3	D3-4	D3-5
Autoleeftijd	<2	≥2	≥2	≥2	≥2
Autoleeftijd	<5	≥5	≥5	<5	≥5
3-jaarschade	≥2.500	≥2.500	<2.500	≥2.500	<2.500
Leeftijd	<70	<70	<70	<70	<70

**Uitsluiten: 2-1 and 3-3**

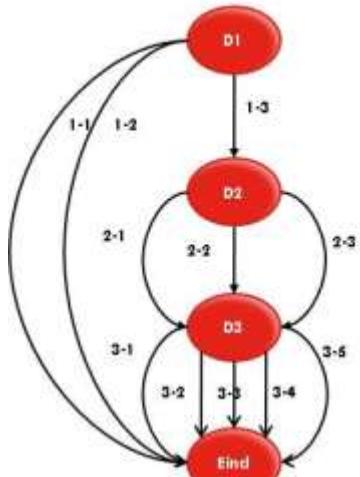


D1	D1-1	D1-2	D1-3
Leeftijd	<18	≥18	≥18
Rijbewijs ingevorderd	N	J	N

D2	D2-1	D2-2	D2-3
Leeftijd	<25	≥25	<25
#jaren rijbewijs	<3	<3	≥3

D3	D3-1	D3-2	D3-3	D3-4	D3-5
Autoleeftijd	<2	≥2	≥2	≥2	≥2
Autoleeftijd	<5	≥5	≥5	<5	≥5
3-jaarschade	≥2.500	≥2.500	<2.500	≥2.500	<2.500
Leeftijd	<70	<70	<70	<70	<70

**Uitsluiten: 2-3 and 3-3**

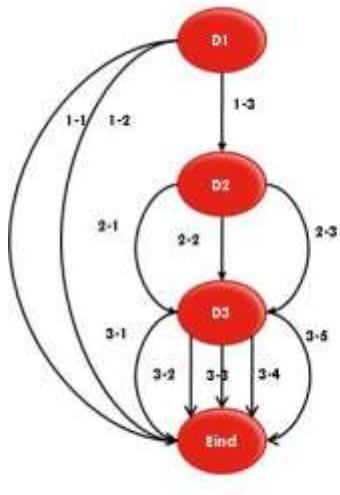


1-1 en 2-2 zijn niet uit te sluiten, aangezien 1-1 direct naar het eind gaat en niet langs 2-2 komt.

D1	D1-1	D1-2	D1-3
Leeftijd	<18	≥18	≥18
Rijbewijs Ingevorderd	N	J	N

D2	D2-1	D2-2	D2-3
Leeftijd	<25	≥25	<25
#jarenrijbewijs	<3	<3	≥3

D3	D3-1	D3-2	D3-3	D3-4	D3-5
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



### Logische testgevallen maken

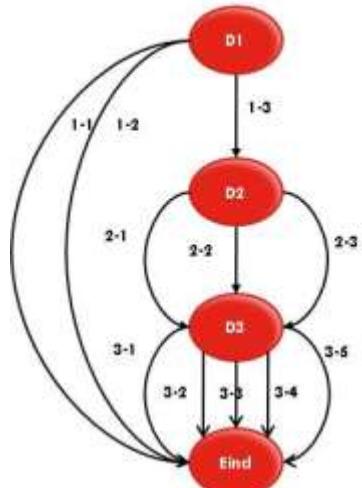
- Bepaal welke testsituaties elkaar uitsluiten
- Combineer met behulp van een matrix

Testsituaties	Waarde	Volgende
D1-1	1	Eind
D1-2	1	Eind
D1-3	0	D2
D2-1	1	D3
D2-2	0	D3
D2-3	0	D3
D3-1	1	Eind
D3-2	1	Eind
D3-3	1	Eind
D3-4	0	Eind
D3-5	0	Eind

### Uitgesloten

D2-1 with D3-3  
D2-3 with D3-3

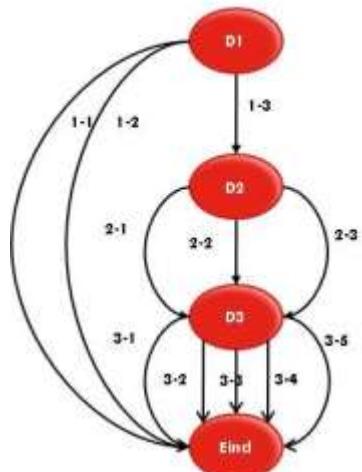
Het resultaat van de testsituatie



Testsituaties	Waarde	Volgorde	TG-1	TG-2	TG-3	TG-4	TG-5	TG-6	TG-7
D1-1	1	Eind	X						
D1-2	1	Eind		X					
D1-3	0	D2							
D2-1	1	D3							
D2-2	0	D3							
D2-3	0	D3							
D3-1	1	Eind							
D3-2	1	Eind							
D3-3	1	Eind							
D3-4	0	Eind							
D3-5	0	Eind							

### Uitgesloten

D2-1 with D3-3  
D2-3 with D3-3



Testsituaties	Waarde	Volgorde	TC-1	TC-2	TC-3	TC-4	TC-5	TC-6	TC-7
D1-1	1	Eind	X						
D1-2	1	Eind		X					
D1-3	0	D2			X	X	X	X	X
D2-1	1	D3			X				
D2-2	0	D3				X			X
D2-3	0	D3				X		X	
D3-1	1	Eind			X				
D3-2	1	Eind				X			
D3-3	1	Eind					X		
D3-4	0	Eind						X	
D3-5	0	Eind							X

### Uitgesloten

D2-1 with D3-3  
D2-3 with D3-3

Testsituaties	TC-4
D1-1	
D1-2	
D1-3	X
D2-1	X
D2-2	
D2-3	
D3-1	X
D3-2	
D3-3	
D3-4	
D3-5	

### Logische testgevallen maken

- Bepaal welke testsituaties elkaar uitsluiten
- Combineer met behulp van een matrix
- Uitbreiden wanneer noodzakelijk

### 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.

Testsituaties	TC-4
D1-1	
D1-2	
D1-3	X
D2-1	X
D2-2	
D2-3	
D3-1	X
D3-2	
D3-3	
D3-4	
D3-5	



Testgeval	TC-4
Testsituaties	D1-3 D2-1 D3-1
Leeftijd	18
Rijbewijs ingetrokken	N
#jaren rijbewijs	2
autoleeftijd	1
3-jaar-schade	2500
Resultaat:	
Foutbericht	-
Premie	2000

D1	D1-1	D1-2	D1-3
Leeftijd	< 18	≥ 18	≥ 18
Rijbewijs ingevorderd	N	Y	N

D2	D2-1	D2-2	D2-3
Leeftijd	< 25	≥ 25	< 25
#jaren rijbewijs	< 3	< 3	≥ 3

D3	D3-1	D3-2	D3-3	D3-4	D3-5
Autoleeftijd	< 2	≥ 2	≥ 2	≥ 2	≥ 2
Autoleeftijd	< 5	≥ 5	≥ 5	< 5	≥ 5
3-jaar-schade	≥ 2500	≥ 2500	< 2500	≥ 2500	< 2500
Leeftijd	< 70	< 70	≥ 70	< 70	< 70

•

#### **4. Vaststellen uitgangssituatie**

Geen bijzonderheden (deze stap is identiek aan de generieke beschrijving van een testontwerptechniek).

#### **5. Opstellen testscript**

Geen bijzonderheden (deze stap is identiek aan de generieke beschrijving van een testontwerptechniek).

### **3.7.8. Gegevenscyclustest (GCT)**

<b>Approach</b>	Dekkingsgebaseerd - Data
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"> <li>• Functionaliteit</li> <li>• Inpasbaarheid</li> <li>• Connectiviteit</li> </ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"> <li>• CRUD</li> <li>• Beslispunten</li> </ul>
<b>Testbasis</b>	<ul style="list-style-type: none"> <li>• CRUD Matrix</li> <li>• Functionele beschrijving</li> </ul>

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 3.5.6) 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 dekkingsvormen 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 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 paragraaf 3.2.2) 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 3.5.6)  
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 geaccoordeerd 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 onderstaande tabel.

	<b>Artikel</b>	<b>Betaling-afspraak</b>	<b>Factuur</b>	<b>Grootboek</b>
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
...	...	...	...	...

Voor dit deel van de CRUD-matrix bestaat één relevante integriteitsregel: Een betalingsafsprak mag niet verwijderd worden als er nog een factuur openstaat met de betreffende betalingsafsprak.

Dit leidt tot twee testsituaties:

IR1-1: Verwijder (D) betalingsafsprak, terwijl er een factuur openstaat met de betreffende betalingsafsprak;

IR1-2: Verwijder (D) betalingsafsprak, zonder dat er een factuur openstaat met de betreffende betalingsafsprak;

Een verkorte overzichtelijke notatie voor dit soort testsituaties is bijvoorbeeld:

<b>Testsituatie</b>	<b>Bewerking</b>	<b>Gegeven</b>	<b>Voorwaarde</b>	<b>Geldig J/N</b>
IR1-1	D	betalingsafsprak	openstaande factuur	N
IR1-2	D	betalingsafsprak	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 3.5.6 en 3.5.7.

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 de twee onderstaande tabellen.

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)

TG-02: "Betalingsafspraak"

Functie	CRUD	Actie / Toelichting
Beheren betalingsafspraken	C	Maak nieuwe betalingsafpraak 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

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 "Betalingsafpraak".

### **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 systeenniveau, 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 4.3, onder "Fase Specificatie": "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 tijdfankelijke gegevens nog geldig zijn en of op basis hiervan wijzigingen in de uitgangssituatie aangebracht moeten worden.

#### **3.7.9. Real Life Test (RLT)**

##### **Kenmerken**

<b>Approach</b>	Dekkingsgebaseerd - Voorkomen
<b>Kwaliteitsattribuut / Testvariëteit</b>	<ul style="list-style-type: none"><li>• Bruikbaarheid</li><li>• Connectiviteit</li><li>• Continuïteit</li><li>• Performance</li></ul>
<b>Dekkingsvorm</b>	<ul style="list-style-type: none"><li>• Operational profiles: opeenvolging van transacties</li><li>• Load profiles: aantallen gebruikers en/of transacties</li></ul>
<b>Testbasis</b>	<ul style="list-style-type: none"><li>• 'Profiles' = beschrijving van het realistisch gebruik</li></ul>

## Beschrijving

Het doel van de real life test is om 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.

Zie paragraaf 3.6.3 en 3.6.4 voor meer informatie.

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 en de testscripts volgens een tevoren vastgelegde

detaillering laten uitvoeren. Maar over het algemeen is dit niet haalbaar. In die gevallen 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.

#### 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 de tabel):

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

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

Omdat het hier niet gaat om het testen van het systeemgebruik in afzonderlijke situaties maakt men hier meestal geen logische testgevallen maar gaat men direct over tot het maken 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

Bij het voorbeeld van de transactieverwerking is voor de betaalautomaattransacties de volgende fysieke invulling gegeven (zie tabel):

Nr	Transactietype	Parameters*	Doel en verwachting
1	Betaalautomaat transacties	Aantal betaaltransacties in 90 minuten op laten lopen van 5 tr/sec naar 450 tr/sec.	Bepalen breekpunt (stresstest). Verwachting: $\geq$ 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.			

#### 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.

#### 5 – Opstellen testscript

Voor het opstellen van het testscript zijn voor deze testontwerptechniek geen speciale aandachtspunten.

## Hoofdstuk 4 TMap NEXT info

### 4.1 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 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 producten zonder het uitvoeren van software.
- Testen : het beoordelen van producten door middel van het uitvoeren van software.

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 25010, 2011). 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 4.4 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.

**Valkulen**

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.

#### **4.1.1. 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 risicotaxatie;
  - 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 systeemontwikkelp producten;
- 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 paragraaf 4.1.3 "TMap in essenties" 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.

## **4.1.2. 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 systeemontwikkelp proces
- Testsoorten en -verantwoordelijkheden
- Testvormen

### **4.1.2.1 Testen en kwaliteitszorg**

Kwaliteit was, is en blijft vooralsnog een uitdaging binnen de automatiseringsbranche. 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 documentatieregels, 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 kwalitedetectie die gebruikt kunnen worden. Kwalitedetectie is op haar beurt slechts één van de activiteiten om kwaliteit te borgen. En kwaliteitsborging is tenslotte slechts één van de dimensies van kwaliteitszorg.

#### 4.1.2.2      **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 18 "De ijsberg").



Figuur 18. 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:

- Explicit testen
- Implicit testen

### **Explicit testen**

Bij explicit testen zijn de testgevallen explicit 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.

### **Implicit testen**

Tijdens het testen kan ook informatie over andere kenmerken (kwaliteitsattributen) worden verzameld, waar niet expliciet testgevallen voor zijn ontworpen. Dit wordt implicit 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 vóór 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.

#### **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.

### **4.1.2.3 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:

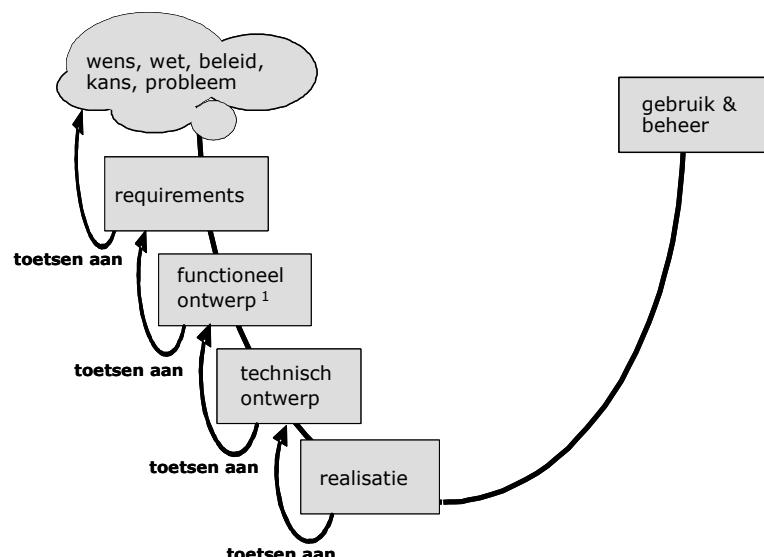
- De plaats van het testen binnen de systeemontwikkelaanpak.
- Bij gebruik van een waterval ontwikkelmethode met onder andere de kenmerken: het in één keer bouwen van het systeem, gefaseerd met duidelijke overdrachtsmomenten, vaak een langdurig cyclisch proces (o.a. SDM).
- Bij het gebruik van een incrementele 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 19 "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.



<sup>1</sup> Met daarin de functionele- en niet-functionele specificaties.

Figuur 19. V-model (de linkerkant).

### Toetsen

Tijdens het systeemontwikkelproces worden diverse tussen- en eindproducten 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 producten in het systeemontwikkelproces zonder het uitvoeren van software.

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?

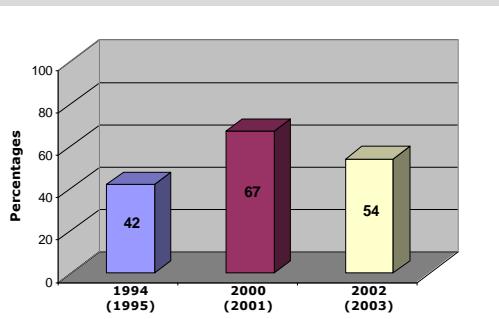
Behalve tussenproducten, kunnen ook eindproducten worden getoetst, bijvoorbeeld door het inspecteren van documentatie zoals beveiligingsprocedures, opleidingen, handleidingen, enzovoort.

Hierbij zijn voor het toetsen diverse technieken beschikbaar: reviews, inspecties en walkthroughs (zie ook hoofdstuk 4.11 "Toetstechnieken").

#### Uitgediept

##### 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 20 "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].



Figuur 20. Gerealiseerde requirements.

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.

### **4.1.3. TMap NEXT® in essenties**

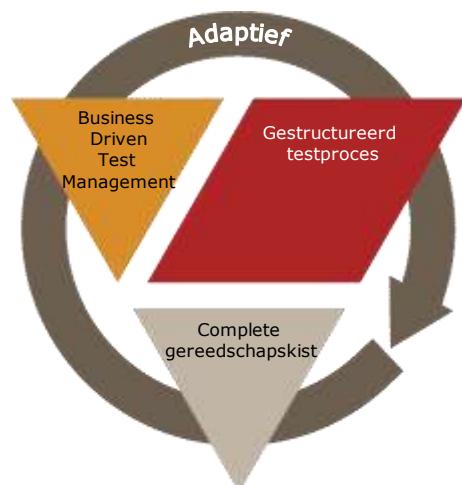
In dit hoofdstuk wordt de specifieke TMap invulling van een gestructureerde testaanpak gegeven. Deze invulling is in een viermaal essenties samenvatten.

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 gereedschapskist.
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 TMap NEXT testproces wordt gebruik gemaakt van het TMap NEXT faseringssmodel. 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 (Building Blocks), zodat het toepasbaar is in diverse systeemontwikkelsituaties: bij zowel traditionele, agile als hybride vormen van systeemontwikkeling, 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 onderstaand "TMap essentiemodel" (zie figuur 21) symboliseert de linker driehoek BDTM, de onderste driehoek de gereedschapskist, de parallellogram het gestructureerde testproces en de 'cirkel' de adaptiviteit van TMap.



Figuur 21. TMap essentiemodel.

#### **4.1.3.1 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 economische 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<sup>2</sup>, 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 explicet 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 risicoadvisie 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 22):

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

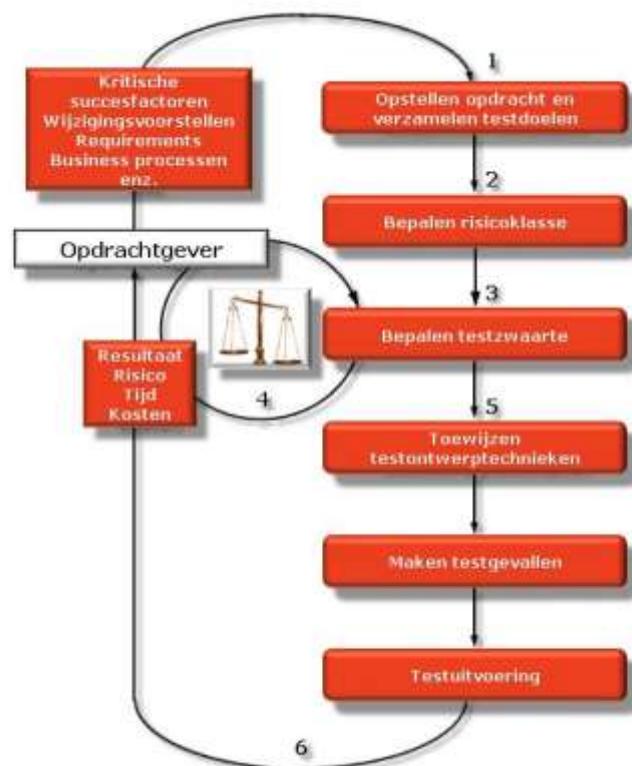
---

2 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.

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 een bepaald welke testsoorten moeten worden ingericht (mastertestplan). Vaak wordt dan al op basis van een productrisicoanalyse<sup>3</sup> bepaald wat er moet worden getest (deelobjecten) en waar naar moet worden gekeken (kenmerken).



Figuur 22. 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").

<sup>3</sup> 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.

Vervolgens wordt in een tabel per combinatie van kenmerk/deelobject en testsoort richting gegeven aan de relatieve diepgang waarmee 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 uitgezet. 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 nastreefd 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 opdrachtgever en andere betrokkenen voldoende inzicht in één 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 stuurbare 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)

### **4.1.3.2 Gestructureerd testproces**

Hier worden de fasering en de activiteiten van volgende TMap NEXT processen beschreven:

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

#### **Masterplan en andere TMap NEXT 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 gereedschapskist, toegelicht (zie de subparagraph 'Complete gereedschapskist').

### **4.1.3.3 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.

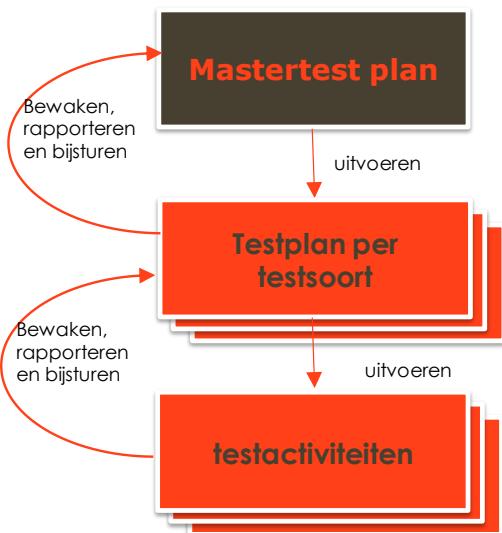
#### *Fase Planning van het totale testproces*

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 "22 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 (zie figuur 23).



Figuur 23. Uitvoeren, bewaken, rapporteren en bijsturen.

### **Proces acceptatie- en systeemtesten**

Zie paragraaf 4.1.5.

### **Proces ontwikkeltesten**

Zie paragraaf 4.1.6.

### **4.1.3.4 Complete gereedschapskist**

Het goed kunnen uitvoeren van het gestructureerde testproces wordt door TMap ondersteund door een complete gereedschapskist. Deze gereedschapskist 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 de TMap Suite op het moment dat ze kunnen worden gebruikt in meer detail beschreven. Met deze gereedschapskist beschikt de tester over een groot aantal mogelijkheden om elke testuitdaging succesvol aan te kunnen gaan.

#### **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 onder meer de volgende onderwerpen:

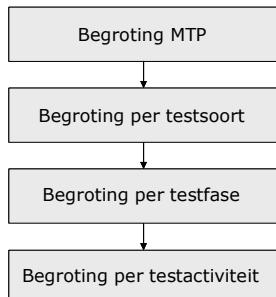
- begroten van de test
- beheren van bevindingen
- opstellen van metrics
- analyseren van productrisico's
- ontwerpen van tests
- toetsen van producten.

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

In het onderstaande worden de (groepen) technieken kort gekarakteriseerd.

### Begroten van de test

Er is een aantal niveaus waarop een begroting kan worden gemaakt. De verschillende niveaus van begroting staan in figuur 24.



Figuur 24. 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. Zie paragraaf 4.7.

## **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**

Zie hoofdstuk 3.

## **Toetsen van producten**

Zie paragraaf 2.20 (Building Block 20: Reviewen van requirements).

## **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](http://www.tmap.net) te vinden en daar te downloaden.

## **Infrastructuur**

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

## **Testomgevingen**

Zie ook paragraaf 4.5.

Voor het 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**

Zie ook paragraaf 2.16 en paragraaf 4.6.

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.

## **Organisatie**

Zie paragraaf 2.4 (Building Block 4: Testorganisatie), paragraaf 2.13 (Building Block 13: Permanente testorganisatie) en paragraaf 2.18 (Building Block 18: Geïntegreerde testorganisatie).

### **4.1.3.5 Adaptieve en complete methode.**

TMap is een aanpak die in alle situaties en in combinatie met elke willekeurige systeemontwikkelmethode toepasbaar is. Het biedt de tester een scala aan elementen voor zijn test, zoals: test approaches, dekkingsvormen, testontwerptechnieken, testinfrastructuur, teststrategie, fasering, testorganisatie, testtools, enzovoort. De tester kiest, afhankelijk van de situatie, die elementen (Building Blocks) 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 veel 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.

#### **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.

## **(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](http://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.

## **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.

## **Probeer voor gebruik**

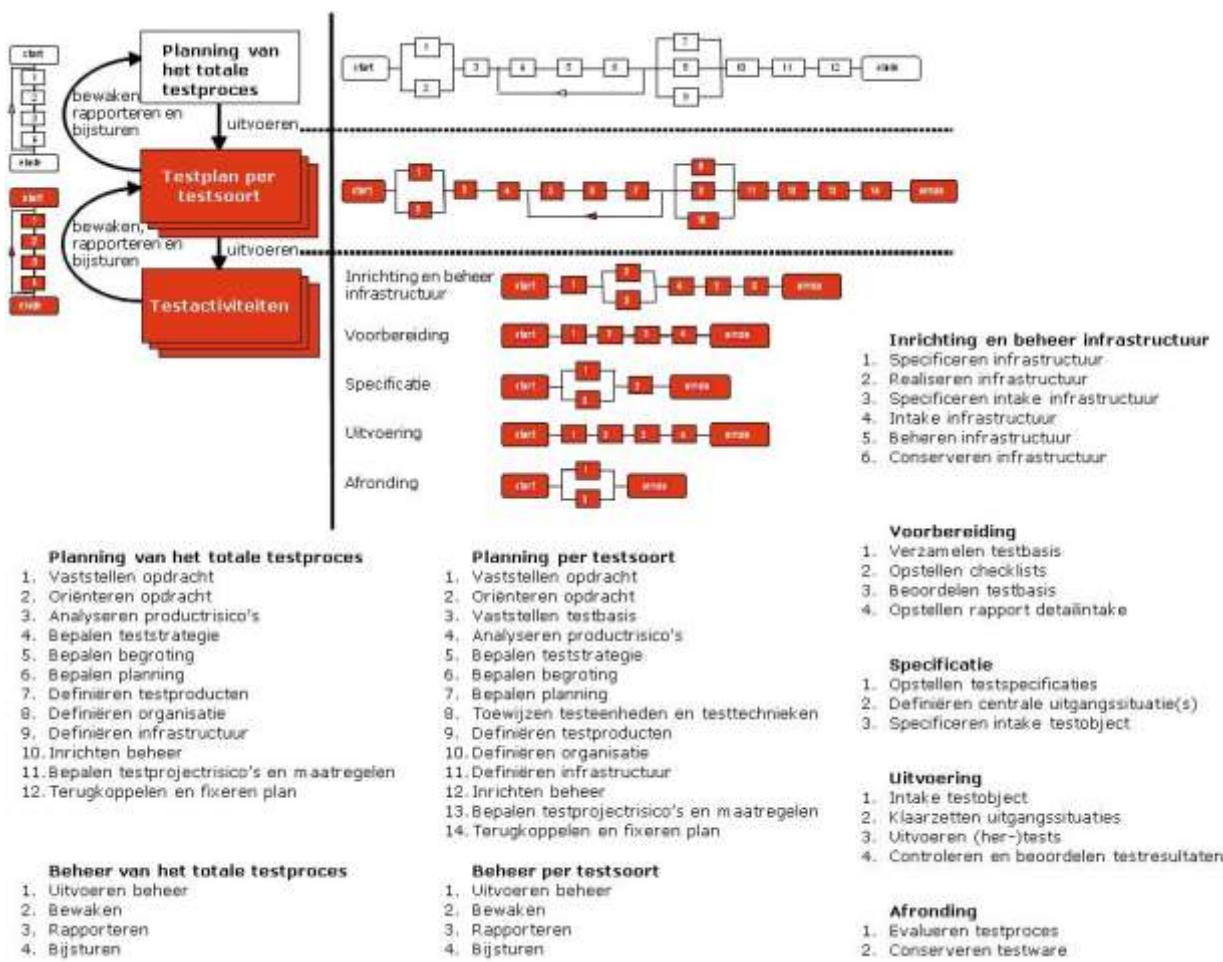
Binnen TMap is ruimte om uit te proberen voordat het werkelijk gebruikt gaat worden. De belangrijkste instrumenten hiervoor zijn de activiteiten rond de intake. De intake van de testbasis, de intake van de testinfrastructuur en de intake van het testobject bieden de ruimte om eerst uit te proberen en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

Het toepassen van TMap betekent niet dat direct alles uit de TMap Suite 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 de TMap Building Blocks. 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. Het specifieke TMap recept voor een bepaalde situatie wordt "patroon" ("pattern") genoemd. TMap nodigt nadrukkelijk uit ook zelf nieuwe patronen te ontwikkelen, op basis van bestaande en/of nieuwe Building Blocks.

### **4.1.4. Testen in een agile omgeving**

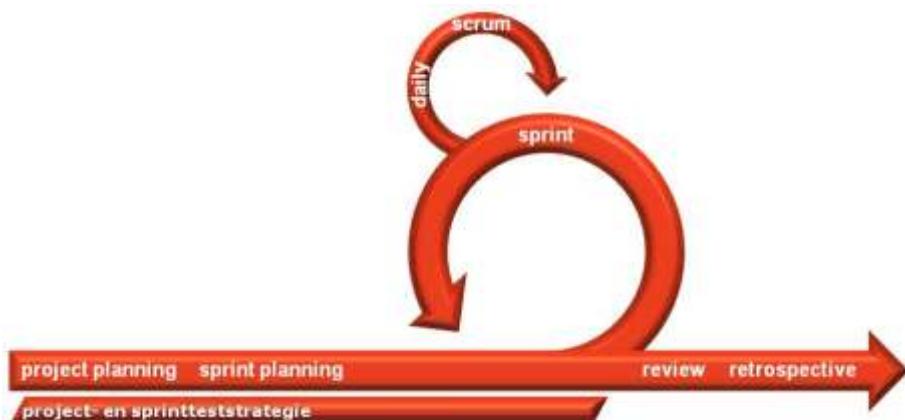
De TMap fasen blijken eenvoudig te integreren met het scrummodel. Maar hoe zit dat met de activiteiten in die fasen? (zie figuur 25) Ook daarvan is in de praktijk gebleken dat deze zonder al te veel problemen aan gebruik in een scrumaanpak zijn aan te passen. In de volgende paragrafen wordt per fase toegelicht hoe de belangrijkste activiteiten in een scrumaanpak kunnen worden uitgevoerd. Houdt bij alles in het achterhoofd, dat scrum een agile en TMap een adaptieve aanpak is. Dit betekent dat alle in de volgende paragrafen genoemde suggesties en praktijkvoorbeelden wellicht aangepast moeten worden op uw situatie.



Figuur 25. TMap NEXT fasen en activiteiten.

#### 4.1.4.1 Planning

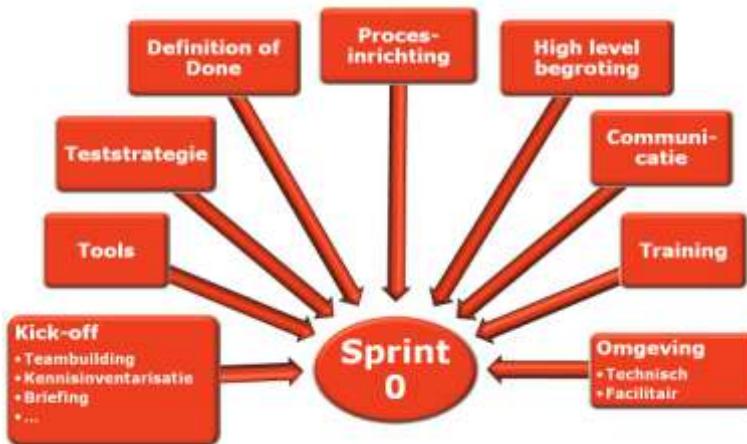
Net als planning in scrum wordt ook de TMap fase planning op diverse momenten uitgevoerd: Aan het begin van de project, aan het begin van iedere sprint en tijdens de daily scrum, zie figuur 26. Het opstellen van een teststrategie is een belangrijke – maar niet de enige – activiteit van de planningsactiviteiten, daarom zijn de momenten van het opstellen van de teststrategie in deze paragraaf apart benoemd; het opstellen van de projectteststrategie bij de start van het project en het opstellen van de sprintteststrategie aan het begin van de sprint. In het kader hiervan wordt in de daily scrum de dagplanning besproken en eventueel aangepast.



Figuur 26. Planningsactiviteiten.

#### 4.1.4.2 Projectteststrategie

De planning van het totale testproces wordt aan het begin van een project opgesteld. Echter het gaat bij scrum niet om het daadwerkelijk plannen (tijdstippen) van taken. Die komen immers op een scrum board te hangen en worden 'actief' op het moment dat het nodig is. Bij scrum gaat het op dit moment van het project vooral om het opstellen van een globale teststrategie. Een projectteststrategie - in een traditionele ontwikkelomgeving zou dit een onderdeel van het mastertestplan zijn - wordt dit ook wel genoemd. Het is goed om te beseffen dat het opstellen van de projectteststrategie tijdens de scrumgebeurtenis "planning" plaatsvindt en in de project planning (product backlog) wordt opgenomen. En in scrumprojecten die starten met een sprint 0, is dit het moment om de projectteststrategie op te stellen. Dit gebeurt dan parallel aan andere sprint 0 'inrichtingsactiviteiten', zoals het houden van een kick-off, inrichten van tools, opstellen definition of done, de procesinrichting, opstellen globale begroting, opstellen communicatieplan en verzorgen van trainingen (zie figuur 27). Deze activiteiten zijn uiteraard slechts voorbeelden van activiteiten die in een sprint 0 kunnen plaatsvinden en kunnen en zullen per organisatie of project anders zijn. Inhoudelijk moet de projectteststrategie compact en globaal zijn. Immers door voortschrijdend inzicht zal de strategie wijzigen tijdens en na de diverse sprints in het project.



Figuur 27. Mogelijke activiteiten in sprint 0.

De belangrijkste onderwerpen om op te nemen in de projectteststrategie, zijn een globale productrisicoanalyse en een globale teststrategie over de sprints heen (zie figuren 28 "Risicotabel" en 29 "Teststrategietabel"). Immers een scrumproject start met de planning. Om dit goed uit te kunnen voeren, is het noodzakelijk inzicht te krijgen in de productrisico's en de gewenste testwaarde. Oftewel de mate waarin de risico's moeten worden afgedekt. Tenslotte zullen hoge risico's en zwaar testen andere consequenties voor de scrumplanning hebben dan lage risico's en licht testen en dit kan gevolgen hebben voor de prioritering van de product backlog items.

Onderwerpen als planning en begroting maken onderdeel uit van genoemde scrumplanning en hoeven weinig tot geen aandacht in de projectteststrategie. De opdracht voor het testen wordt door de product owner gegeven en kan worden vertaald in bijvoorbeeld eisen, die in de definition of done worden opgenomen. Dit behoeft dus ook nauwelijks aandacht in de projectteststrategie.

Het definiëren van de testinfrastructuur (onder andere testomgevingen, testdata en testtools) zou nog in een plan opgenomen kunnen worden, maar als in een sprint 0 is voorzien, is het beter de testinfrastructuur direct in te richten en gereed te hebben, of in ieder geval hiermee een start te maken, voordat de eerste sprint start. Is er geen sprint 0 voorzien, dan kunnen deze taken ook als technische product backlog items op de backlog worden opgenomen.

#### **4.1.4.3 Sprintteststrategie**

De sprintteststrategie wordt tijdens de sprint planning definitief gemaakt en in de sprint planning (sprint backlog) opgenomen. In de sprint planning wordt bijvoorbeeld met planningpoker door het ontwikkelteam per sprint backlog item het aantal uren geschat. Het aantal uren wordt beïnvloed door het wel of niet zwaar testen en dat heeft weer een relatie met het productrisico. Het verdient daarom aanbeveling om op een backlog item - met name bij user stories -, naast de door de product owner bepaalde prioriteit, ook de risicoklasse op te nemen voordat met planningpoker wordt gestart.

De sprint test strategie is een detailinvulling van de projectteststrategie en kan gedurende de sprint wijzigen. Net als bij de projectteststrategie geldt hier dat het een compacte strategie moet zijn. Het zou bij voorkeur op een whiteboard moeten passen. Het belangrijkste onderdeel is de productrisicoanalyse en teststrategie voor de huidige sprint. Het team is de belangrijkste bron voor het uitvoeren van de productrisicoanalyse.

De in de TMap beschreven risicoanalyse kan als volgt worden aangepast en uitgevoerd. Uiteraard is het gehele scrumteam hierbij aanwezig en doet actief mee. Aangezien de analyse per sprint plaatsvindt, gaat het niet om echt grote aantallen backlog items, zodat de analyse niet al te lang hoeft te duren. Een uur blijkt vaak ruim voldoende te zijn. Het uitvoeren kan eenvoudig met de volgende stappen worden gedaan:

1. Verzamel de scrumteamleden.
2. Schrijf alle backlog items van de huidige sprint onder elkaar op een whiteboard.
3. Vraag aan alle teamleden welke kwaliteitsattributen per backlog item voor ieder individueel teamlid belangrijk is.
4. Bepaal per combinatie van backlog item en kwaliteitsattribuut wat de schade en faalkans is. Het productrisico is dan: schade x faalkans.

De risicotabel zou er dan zo uit kunnen zien (zie figuur 28).

Item	Kenmerk	Schade	Faal-kans	Risico-klasse
A	Functionaliteit	3	3	9
	Usability	2	1	2
B	Functionaliteit	2	2	4
	Beveiliging	3	2	6
C	Functionaliteit	2	1	2
D	Performance	2	1	2
E	Performance	1	1	1
F	Functionaliteit	2	2	4
	Inpasbaarheid	2	2	4
**	**	..	..	..

Figuur 28. Risicotabel.

In de volgende stap moet dan de teststrategie worden bepaald. Hierin wordt bepaald met welke testzwaarte een combinatie van backlog item en kwaliteitsattribuut moet worden getest. Om het praktisch toepasbaar te maken, kunnen er simpelweg kolommen als "testzwaarte" en "testontwerptechniek" worden toegevoegd. Dit helpt de teamleden met een testrol bij het maken van de testgevallen (zie figuur 29). Pas 'simpelweg', per regel in de tabel, de genoemde testontwerptechniek toe op het backlog item en de testzwaarte - testdiepgang - in relatie met het af te dekken risico wordt bereikt.

5. Bepaal de testzwaarte.
6. Bepaal op basis van testzwaarte en kwaliteitsattribuut, welke testontwerptechniek moet worden toegepast.

Item	Kenmerk	Schade	Faal-kans	Risico-klasse	Zwaarte	Test-onwerp-techniek
A	Functionaliteit	3	3	9	***	EVT-MCC
	Usability	2	1	2	*	SYN
B	Functionaliteit	2	2	4	**	ET
	Beveiliging	3	2	6	**	SEM-MCDC
C	Functionaliteit	2	1	2	*	DCT-EQ
D	Performance	2	1	2	*	EG
E	Performance	1	1	1	*	EG
F	Functionaliteit	2	2	4	**	EVT-MCDC
	Inpasbaarheid	2	2	4	**	PCT-TM2
**	**	..	..	..	..	..

Figuur 29. Teststrategietabel.

- ET : Exploratory Testing  
 EVT-MCC : Elementaire vergelijkingstest - Multiple Condition Coverage,  
 SYN : Syntactischetest,  
 EVT-MCDC : Elementaire vergelijkingstest - Modified Condition/Decision Coverage,  
 SEM-MCDC : Semantischetest - Modified Condition/Decision Coverage,  
 DCT-EQ : Datacombinatietest - Equivalentieklassen,  
 EG : Error Guessing,  
 PCT-TM2 : Procescyclustest - Testmaat 2  
 Meer weten? Kijk eens in TMap NEXT [TMapNEXT, 2006].

In de teststrategietabel (figuur 29) staan nu testontwerptechnieken en dekkingsvormen vermeld. Ook de test approaches kunnen vermeld worden. Er zijn daarnaast natuurlijk ook andere kwaliteitsmaatregelen mogelijk, zoals toetsen (bijvoorbeeld het uitvoeren van een review of een inspectie), pairprogramming of het uitvoeren van een zwaardere unittest. U kunt de tabel compleet aanpassen aan uw situatie. Dus wilt u de kolom "Testontwerptechniek" ruimer van begrip maken? Dan wijzigt u die kolom toch gewoon in bijvoorbeeld "Kwaliteitsmaatregelen". Ook een kolom "Moment/Locatie" wordt wel toegevoegd.

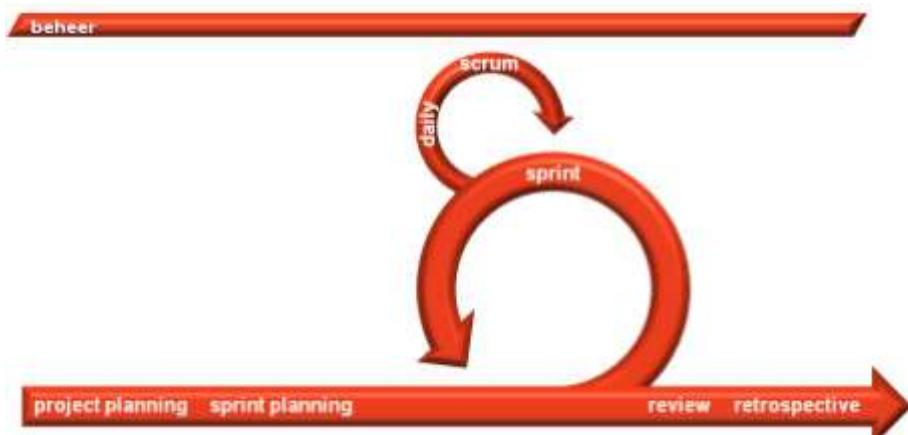
Nu de teststrategietabel gereed is, is het zaak de regels in de tabel als taken op te nemen op het scrum board. De teststrategietabel blijft dus geen 'losstaande' tabel!

Het is niet noodzakelijk een onderscheid te maken tussen de diverse testsoorten. Maar mocht dit toch gewenst zijn dan kunnen bijvoorbeeld alle user stories waarbij kwaliteitsattribuut "functionaliteit" is genoemd, worden gegroepeerd tot een systeemtest en alle user stories met bijvoorbeeld "usability" en "inpasbaarheid" tot een acceptatietest. En deze kunnen dan tijdens of parallel aan de sprint, of daarna worden uitgevoerd.

De teststrategie staat aan de basis van iedere testactie, activiteit, proces of project. Het bevat de legitimatie over wat en hoe er getest moet worden, welke risico's er zijn en welke wel en hoe, of niet, afgedekt moeten worden. Dit kan prioriteiten in een scrumproject beïnvloeden. Meer risico, hogere prioriteit bijvoorbeeld. Ook beslissingen in wat wel of niet te doen met het oog op tijd, kosten en resultaat (kwaliteit) kunnen hierdoor worden beïnvloed. Kortom, een goed doordachte teststrategie faciliteert het gehele scrumteam.

#### **4.1.4.4 Beheer**

In scrum is de TMap fase beheer meer een faciliterende dan een controlerende activiteit. De teamleden moeten elkaar vertrouwen en op hun beurt worden vertrouwd door het management. Een onderdeel van beheer is de 'daily scrum', waarin de voortgang van de testactiviteiten wordt gerapporteerd en inzichtelijk gemaakt. Tevens wordt hier de dagplanning besproken en eventueel aangepast (figuur 30).



Figuur 30. Beheeractiviteiten.

#### 4.1.4.5 Voortgang

Een pragmatische manier om de voortgang inzichtelijk te maken, is het uitbreiden van de teststrategietabel, met enkele kolommen: "Tests gemaakt J/N", "Tests uitgevoerd J/N" en "Tests passed J/N".

Item	Kenmerk	S K	F K	R K	Zwaarte	Test- ontwerp- techniek	Tests gemaakt (J/N)	Tests uitgevoerd (J/N)	Tests passed (J/N)
A	Functionaliteit	3	3	9	***	EVT-MCC			
	Usability	2	1	2	*	SYN			
B	Functionaliteit	2	2	4	**	ET			
	Beveiliging	3	2	6	**	SEM-MCDC			
C	Functionaliteit	2	1	2	*	DCT-EQ			
D	Performance	2	1	2	*	EG			
E	Performance	1	1	1	*	EG			
F	Functionaliteit	2	2	4	**	EVT-MCDC			
	Inpastbaarheid	2	2	4	**	PCT-TM2			
...	...	-	-	-					

Figuur 31. Testvoortgangstabel.

Met deze tabel (figuur 31) op het whiteboard ziet iedereen in één oogopslag wat de testvoortgang is. Of, als de regels van de teststrategietabel zijn vertaald naar taken op de sprint backlog; op het scrum board ziet iedereen in één oogopslag wat de (test)voortgang is. Tenslotte staan op het scrum board de taken in de kolom met de actuele status; to do, in progress of done. Verder geven zowel de product als sprint burndown charts ook informatie over de actuele status en voortgang en kunnen op basis daarvan bijsturingen plaatsvinden.

#### Bevindingen

Tijdens de sprint wordt ook over bevindingen gecommuniceerd en eventuele te nemen maatregelen bediscussieerd en ingevoerd. Bevindingen die in een sprint kunnen worden opgelost, worden niet geregistreerd. In dit geval gaan vaak de tester en de ontwikkelaar samen zitten om de bevinding - snel - op te lossen. Om ervoor te waken dat het oplossen van bevindingen hoofdactiviteit wordt in een sprint is het goed onderstaande richtlijnen in acht te nemen. Een bevinding wordt in de bevindingenadministratie opgenomen als:

- **de bevinding niet in één dag kan worden opgelost,**
- **het team besluit om - in overleg met de product owner - in een andere sprint de bevindingen op te lossen,**
- **een tijdens de sprint review gevonden bevinding, niet in de sprint review kan worden opgelost.**

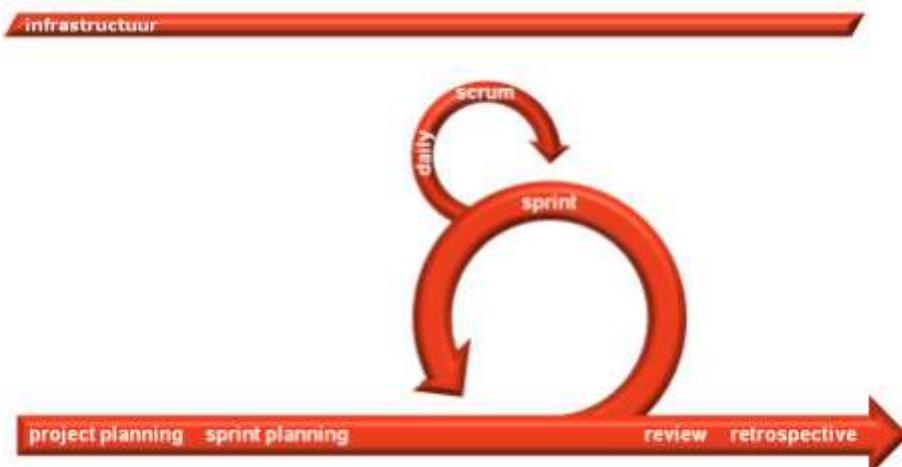
Uiteraard kan het team hier van afwijken. Als het team vindt dat er meer dan het minimum moet worden geregistreerd is dat prima. Bijvoorbeeld als er 'betrouwbare' metrieken moeten worden opgebouwd. Dit moet dan wel in de definition of done zijn opgenomen.

Voor het opnemen en afhandelen van bevindingen, kunnen 'standaard' bevindingenprocedures worden gebruikt. Er is echter één groot verschil. Niet de testmanager maar het scrumteam is verantwoordelijk voor het registreren en monitoren

van de bevindingen. In het geval een bevinding de scope van het scrumteam overstijgt, dan kan deze worden toegekend aan de scrum master.

#### 4.1.4.6 Inrichting en beheer infrastructuur

Zoals al eerder beschreven, is het een goede zaak om al te starten met het inrichten van de testinfrastructuur in een sprint 0 (figuur 32).



Figuur 32. Inrichting en beheer infrastructuuractiviteiten.

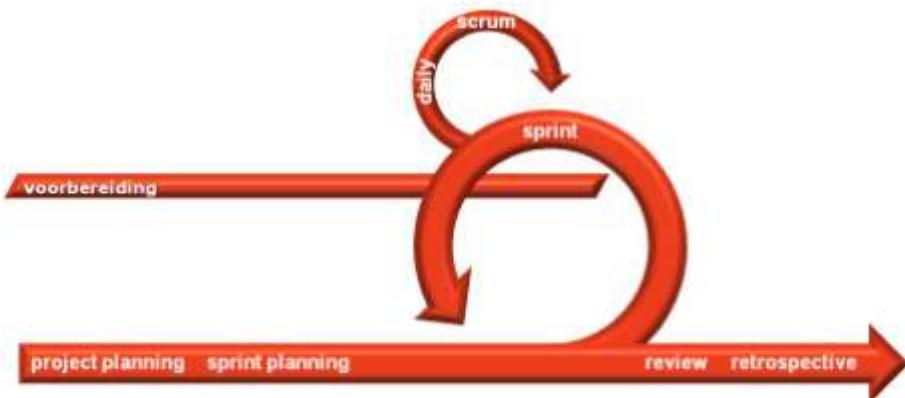
De testinfrastructuur bestaat uit onder andere testomgevingen,testdata en testtools. Het beheer hiervan vindt gedurende het gehele scrumproject plaats. Vaak wordt verwacht dat het scrumteam de testinfrastructuur zelf inricht en onderhoudt. Dit betekent dat in die situatie er genoeg teamleden moeten zijn met voldoende technische kennis om dit in te kunnen vullen.

Ook in de situatie, of misschien zelfs wel, juist in de situatie dat het inrichten van de testinfrastructuur buiten het team plaatsvindt, is het noodzaak daar zo vroeg mogelijk actie op te nemen. Immers de testinfrastructuur moet stabiel zijn, gezien de korte doorlooptijden van een sprint en dus ook de korte periode waarin getest moet worden.

Het zal duidelijk zijn, dat problemen in de testinfrastructuur grote invloed op de voortgang van activiteiten in een sprint kunnen hebben.

#### 4.1.4.7 Voorbereiding

Het toetsen van de testbasis is een activiteit die per product backlog item wordt uitgevoerd en al start aan het begin van het project. Het toetsen van de product backlog items vindt tevens parallel plaats aan de ontwikkeling van andere product backlog items.



Figuur 33. Voorbereidingsactiviteiten.

Door per product backlog item toelichting te vragen of kritische vragen te stellen aan de product owner, de gebruiker (bezit materiekennis) en de ontwikkelaar (bezit technische kennis) krijgt de tester inzicht in hetgeen met het product backlog item bereikt moet worden. Uiteraard betreft dit een wisselwerking, want door de kritische vragen worden de andere teamleden in staat gesteld hun producten kwalitatief te verbeteren. Uiteraard hoeft niet alles te worden gedocumenteerd, als het maar wel wordt gecommuniceerd, zodat er geen informatie verloren gaat.

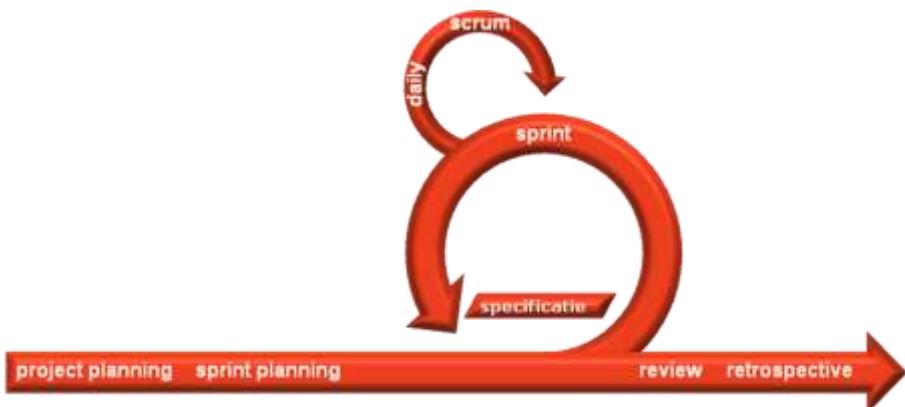
Als de tester genoeg informatie heeft om testgevallen te kunnen specificeren, is de toets - fase voorbereiding - van het product backlog item afgerond. Het schrijven van een rapport detailintake is in deze situatie overbodig, omdat alle betrokkenen al op de hoogte zijn en eventuele maatregelen al zijn getroffen. Als er in definition of done aanvullende eisen zijn opgenomen, dan moet daar uiteraard aan worden voldaan.

Als een ander teamlid - niet zijnde de tester - een testrol krijgt, kan het nuttig zijn om dit teamlid een checklist te geven, waarmee de toets kan worden uitgevoerd. Deze checklist bevat vragen over compleet en consistent zijn van het product backlog item, het kunnen toepassen van de gekozen testontwerptechniek, enzovoort. Hiermee kan de kwaliteit en uniformiteit van de toets worden geborgd.

Naar aanleiding van de toetsresultaten is het mogelijk dat de productrisicoanalyse en teststrategie worden aangepast of dat er andere approaches, dekkingsvormen en /of testontwerptechnieken worden gekozen.

#### **4.1.4.8 Specificatie**

Het specificeren van de testgevallen, is een continue activiteit die per product backlog item wordt uitgevoerd. Vaak al tijdens het toetsen van de product backlog items. Het specificeren vindt tevens parallel plaats aan de ontwikkeling van andere product backlog items (figuur 34).



Figuur 34. Specificatieactiviteiten.

Concreet betekent dit, als er een teststrategietabel (figuur 29) is opgesteld, er per regel met de genoemde testontwerptechniek de testgevallen voor het desbetreffende product backlog item worden gemaakt.

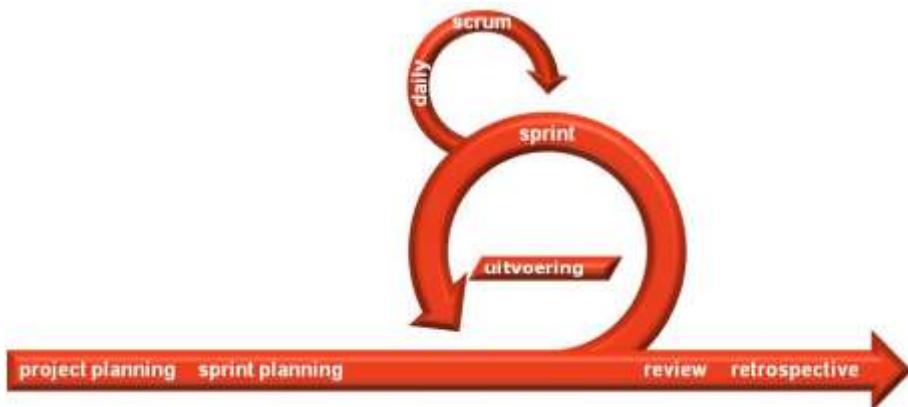
Met welke diepgang de testgevallen gedocumenteerd moeten worden, kan afhangen van de door de organisatie gestelde eisen in verband met overdraagbaarheid of van wet- en regelgeving. Dit is dan vastgelegd in de definition of done. In alle andere gevallen, moet de documentatie die mate van diepgang hebben, waarmee de tester de tests kan uitvoeren.

Gezien de korte doorlooptijd van een sprint, verdient het aanbeveling om de tests direct te automatiseren, of in ieder geval zo op te stellen, dat ze geautomatiseerd kunnen worden. Unitests worden bijna altijd geautomatiseerd uitgevoerd - dit kan in een definition of done zijn opgenomen - en om zowel van de techniek als van het product kennis te nemen, is het goed als de tester hierbij betrokken is.

Tijdens het toetsen krijgt de tester vaak heel veel productinformatie van de product owner, gebruiker en ontwikkelaar, die wel – direct – in de testgevallen wordt verwerkt, maar niet altijd in de product backlog items zelf of andere ontwerpproducten worden opgenomen. In de praktijk blijkt dan ook regelmatig dat de testgevallen meer productinformatie bevatten dan de oorspronkelijk product backlog items. Hiermee vormen de testgevallen een waardevolle kennisbron, waar in volgende sprints en projecten veel voordeel mee kan worden behaald. Zorg er, zeker in deze situatie, wel voor dat het conserveren van testware in de definition of done is opgenomen.

#### **4.1.4.9 Uitvoering**

Het uitvoeren van de testgevallen wordt vaak parallel aan andere product backlog items uitgevoerd (figuur 35).



Figuur 35. Uitvoeringsactiviteiten.

Het uitvoeren van de pretest is meestal beperkt van omvang of zelfs in het geheel niet nodig. Immers de tester is aanwezig in het team, kijkt mee met de unittests en weet dus exact wat er aan komt. Door het ontbreken van overdrachtsmomenten, naar bijvoorbeeld onafhankelijke testteams, is het uitvoeren van een pretest ook niet nodig.

Tijdens het uitvoeren van de tests worden bevindingen gedaan.

#### **4.1.4.10 Test Driven Development**

Scrumprojecten gebruiken vaak een Test-Driven Development (TDD) aanpak. Dit is een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna pas de code. Hoewel TDD eigenlijk meer valt onder eXtreme Programming dan onder scrum, wordt het veel in scrum gebruikt. Enkele voordelen van TDD zijn: Er wordt gekeken vanuit het perspectief van de gebruiker. De testgevallen waarvoor de code wordt geschreven zijn namelijk gebaseerd op de backlog items, die vanuit het oogpunt van de product owner zijn opgesteld. Doordat alle code van het begin af aan wordt getest, wekt dit meer vertrouwen bij de product owner.

Waar voordelen zijn, zijn ook nadelen: De programmeur schrijft bij TDD zowel de (unit)tests als de code voor de applicatie. Dit heeft tot gevolg dat wanneer de programmeur iets over het hoofd ziet, dit in zowel de test als in de code over het hoofd gezien zal worden. Een valkuil die in de praktijk voor komt, is het niet of nauwelijks uitvoeren van een integratie- en systeemtest.

#### **4.1.4.11 Testautomatisering**

Zoals eerder geschreven verdient het aanbeveling de tests geautomatiseerd uit te voeren. Dit geldt zeker voor unit- en regressietests. Maar ook de andere tests komen hiervoor in aanmerking, bijvoorbeeld ook als onderdeel van een continuous build en integratiestrategieën. Zoals bij heel veel aspecten, dient het wel of niet ontwikkelen en uitvoeren van een geautomatiseerde test in de 'definition of done' te zijn opgenomen.

Voor het automatiseren zelf worden in de praktijk diverse oplossingen gekozen. Soms in de sprint zelf door de teamleden zelf, soms parallel aan de sprint door anderen en een aanpak die in de praktijk ook wel plaatsvindt, gaat als volgt. De tijdens de sprint opgestelde en handmatig uitgevoerde testgevallen worden parallel aan de volgende sprint geautomatiseerd en in een regressietest opgenomen. In de sprint daarna kan dan de regressietest geautomatiseerd worden uitgevoerd. Dit gebeurt dan naast het uitvoeren van de handmatige testgevallen van desbetreffende sprint. En zo verder. Aan het eind

van het project - of er net na - is dan een volledig geautomatiseerde regressietest beschikbaar. Ook hier geldt dat dit dan onderdeel van de definition of done moet zijn.

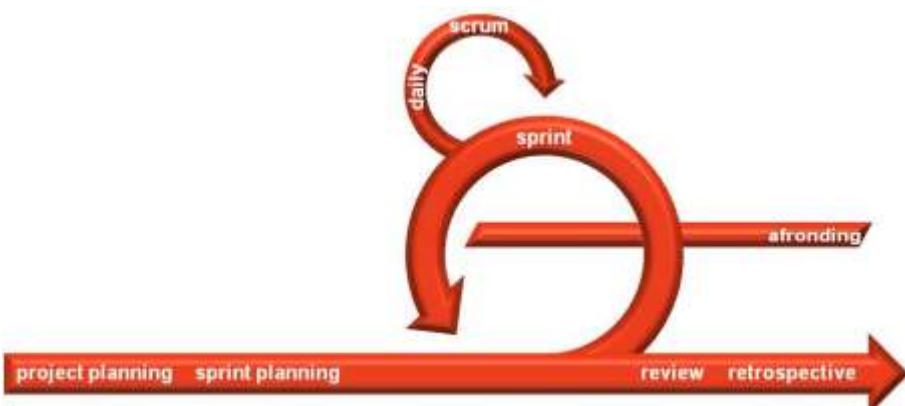
#### 4.1.4.12 Afronding

Het evalueren van het testproces past perfect in de (sprint) retrospective, waarna verbeteringssuggesties zoveel als mogelijk in de volgende sprint doorgevoerd moeten worden.

Het conserveren van de testware vindt plaats aan het eind van de sprint of aan het eind van het project (figuur 36). Wat en hoeveel testware geconserveerd moet worden en of er een configuratiemanagementtool moet worden gebruikt, is vastgelegd in de definition of done.

Een scrumproject bestaat vaak uit meer dan één sprint. Het is dan ook belangrijk aandacht te besteden aan het opbouwen van een regressietestset. Dit kan bijvoorbeeld door de belangrijkste testgevallen uit de huidige sprint op te nemen in een regressietestset. Welke de belangrijkste zijn, kan worden vastgesteld met behulp van de risicotabel (figuur 28). Deze omvang van deze set groeit per sprint. En als dan in een volgende sprint, naast het testen van dan actuele items ook een regressietest moet worden uitgevoerd, dan ontbreekt vaak de tijd daarvoor. Het is dan ook goed, om bij het samenstellen van een regressietestset, na te denken over hoe de regressietest geautomatiseerd kan worden uitgevoerd.

'Responding to change' is een agile waarde, die niet alleen voor wijzigingen in backlog items geldt, maar bijvoorbeeld ook voor wijzigingen in rollen en teamsamenstellingen. Dit reageren op veranderingen, in bijvoorbeeld testrollen, kan alleen goed worden uitgevoerd als voldoende aandacht wordt geschonken aan onderhoudbare, overdraagbare en herbruikbare testware. In deze situatie is testwaremanagement een belangrijk hulpmiddel. En ook hier geldt weer, dat als dat belangrijk is, dat dan in de definition of done moet zijn opgenomen.



Figuur 36. Afrondingsactiviteiten.

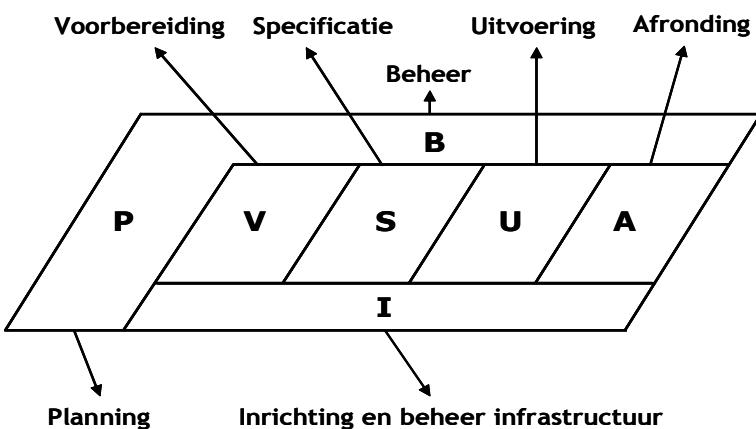
#### 4.1.5. Proces acceptatie en systeemtesten

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 faseringssmodel 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 faseringmodellen voor systeemontwikkeling te hanteren. In het faseringmodel van TMap zijn de testactiviteiten verdeeld over een zevental fasen: Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding (zie figuur 37 "TMap faseringmodel"). 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 37. 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 faseringssmodel (parallelogram) 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 beheerde 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.

#### **4.1.6. 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 faseringssmodel 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 productkwaliteit/kwaliteit 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 ontwikkelen 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 ontwikkelen zijn het meestal de ontwikkelaars zelf die testen.

## **4.2 Testprofessionals**

### **4.2.1. 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. Hieronder, in paragraaf "Aandachtspunten" worden een aantal punten besproken die aandacht verdienen bij de inrichting van HRM voor testprofessionals. De paragraaf "Eigenschappen" beschrijft wat een tester nou een tester maakt. Wat zijn bijvoorbeeld de persoonlijke kenmerken van een tester?

### **4.2.2. 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 niets anders meer is. Hier volgen een aantal aandachtspunten voor de inrichting van HRM bij testprofessionals.

#### **4.2.2.1 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 gevuld kunnen worden. Voor de testers ontbreekt dat soms met bijvoorbeeld het excus 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 paragraaf 2.6 "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 betrokken te laten beoordelen met daadwerkelijke testervaring. Bijvoorbeeld de testcoördinator 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 veroorzaaker 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).

### 4.2.3. 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

discusses 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.

## **4.3 Acceptatie en systeemtesten**

### **4.3.1. Inleiding**

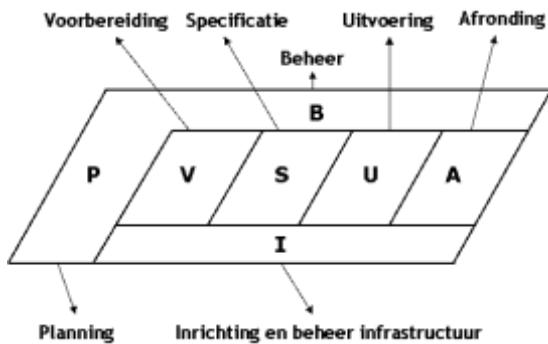
#### **Acceptatietest en systeemtest**

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.

In een ontwikkelproces kan 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 38 "TMap faseringsmodel"). Dit zijn de fasen Planning, Beheer, Inrichting en beheer infrastructuur, Voorbereiding, Specificatie, Uitvoering en Afronding.



Figuur 38. TMap faseringenmodel.

In de fase Planning 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 worden de activiteiten uit het testplan uitgevoerd, bewaakt en eventueel bijgestuurd. De fase Inrichting en beheer infrastructuur heeft als doel om zorg te dragen voor de benodigde testinfrastructuur die wordt gebruikt bij de verschillende TMap fasen en activiteiten. De fase Voorbereiding 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 en uitgevoerd in de fase Uitvoering. Zo wordt inzicht verkregen in de kwaliteit van het testobject. Tijdens de fase Afronding 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.

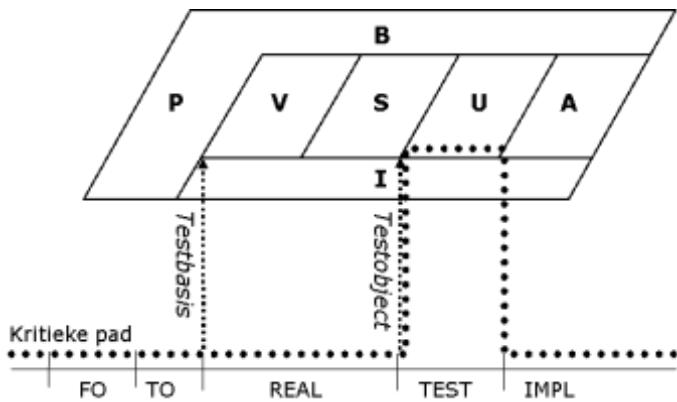
De hiervoor beschreven fasen hoeven niet altijd geheel sequentieel te worden uitgevoerd. Het is bijvoorbeeld heel goed mogelijk dat testgevallen voor een deel van de test nog gespecificeerd worden (fase Specificatie), terwijl de testuitvoering (fase Uitvoering) voor een ander deel van de test al is gestart. 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 faseringenmodel 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 faseringenmodel**

Binnen het faseringenmodel 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 faseringenmodel 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 39 "Relatie TMap testfasering met systeemontwikkelfasering" is een voorbeeld van deze relaties weergegeven.



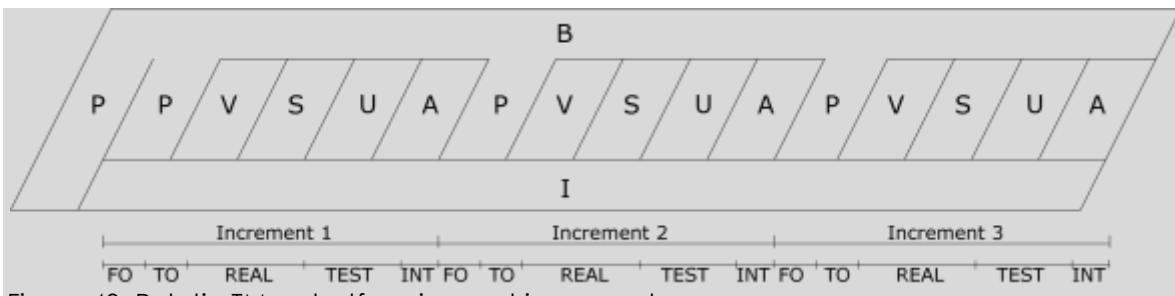
Figuur 39. Relatie TMap testfasering met systeemontwikkelfasering.

In figuur 39 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 systeemontwikkelfasering kent zoals die in figuur 39 "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 geïntegreerd (INT). In figuur 40 "Relatie TMap testfasering met incrementen" wordt dat schematisch weergegeven.



Figuur 40. 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.

### 4.3.2. Fase Planning

#### Doele

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 "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.

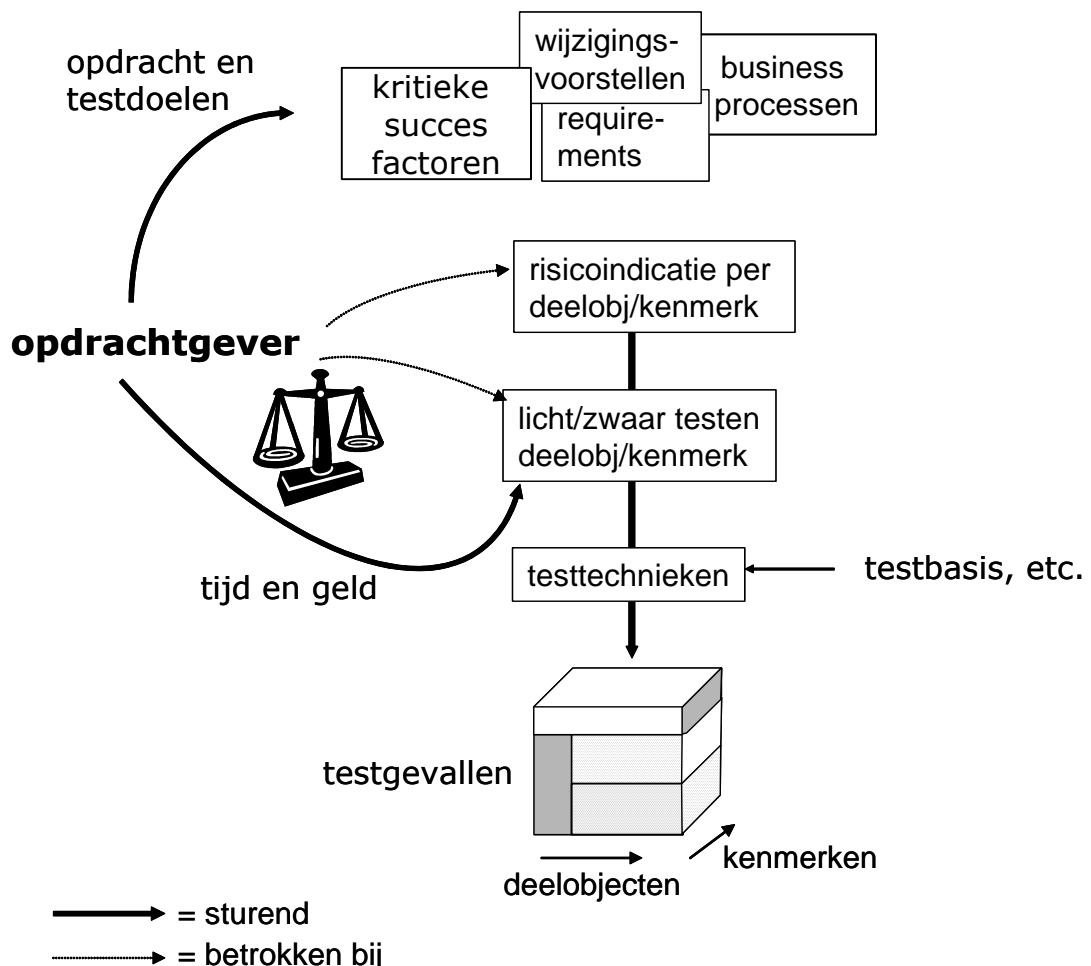
Indien vanuit het mastertestplan nog geen productrisicoanalyse is uitgevoerd of deze te globaal is, wordt samen met de opdrachtgever en andere betrokkenen een gedetailleerde analyse uitgevoerd om op basis van de opdracht vast te stellen wat de gewenste resultaten van het testen voor de opdrachtgever 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 41 maakt dit duidelijk.



Figuur 41. 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 testcoördinator 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:

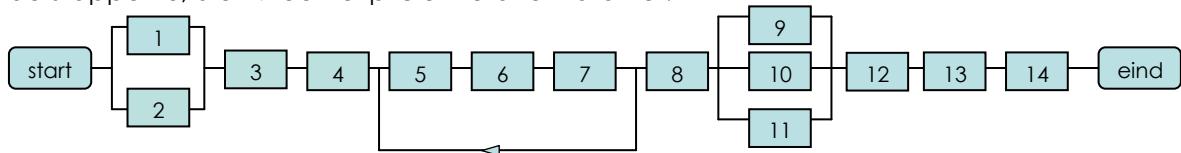
- Hoe meer bevoegdheden, hoe meer de voorkeur uitgaat naar de term testmanager;
- Hoe groter de scope van de test, idem;
- Hoe groter de omvang van de test, idem;
- Bij een overkoepelende testmanager: voorkeur testcoördinator;
- Bij een overall testcoördinator: voorkeur testmanager;

### **Activiteiten**

Het opstellen van het testplan omvat de volgende activiteiten:

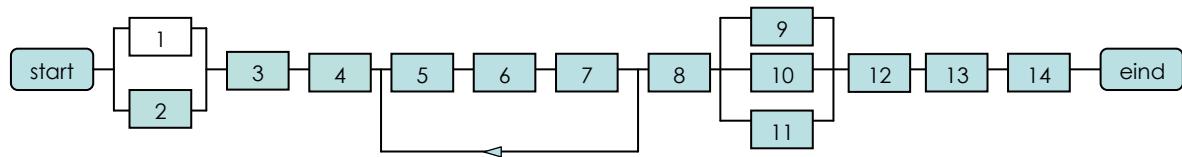
1. Vaststellen opdracht;
2. Oriënteren opdracht;
3. Vaststellen testbasis;
4. Analyseren productrisico's;
5. Bepalen teststrategie;
6. Bepalen begroting;
7. Bepalen planning;
8. Toewijzen testeenheden en testtechnieken;
9. Definiëren testproducten;
10. Definiëren organisatie;
11. Definiëren infrastructuur;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

Onderstaand schema (zie figuur 42) 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 42. Opstellen testplan.

### 4.3.2.1 Vaststellen opdracht



#### Doele

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.

### **Uitgediept**

#### 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 ¾ 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
- 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 behoeft te hebben aan materiedeskundige ondersteuning, en geldt de behoeft 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 draagt 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 wens te 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](http://www.tmap.net).

### Producten

De opdrachtformulering, vastgelegd in het testplan.

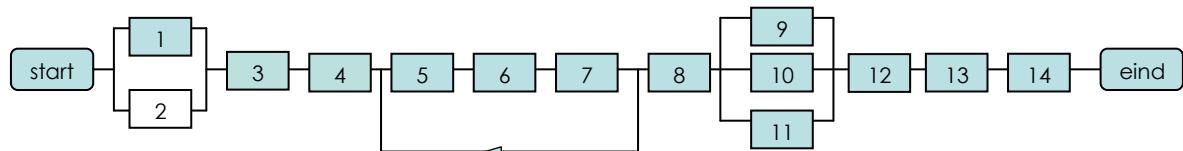
### Technieken

Checklist randvoorwaarden en uitgangspunten ([www.tmap.net](http://www.tmap.net)).

### Tools

Niet van toepassing.

## 4.3.2.2 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 gevuld 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 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 functiepuntonalyse; 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 systeemontwikkelproces worden geïnterviewd. Voor de interviews is een checklist beschikbaar op [www.tmap.net](http://www.tmap.net).

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 (of 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 Gebruiks frequentie). 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 betrokkenen 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 gevuld (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.

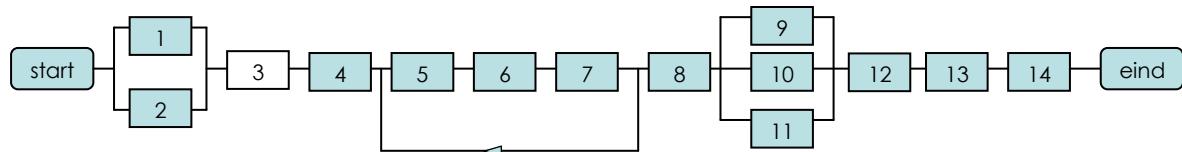
## **Technieken**

Checklist "Oriënteren testopdracht" ([www.tmap.net](http://www.tmap.net)).

## **Tools**

Niet van toepassing.

### **4.3.2.3 Vaststellen testbasis**



## **Doe**

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.

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 productieisen 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 unit-test 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.

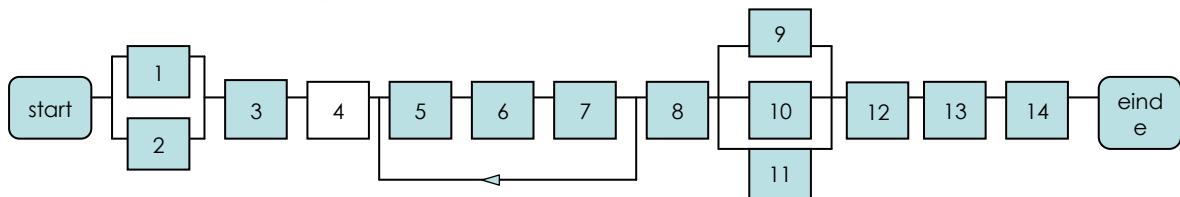
## **Technieken**

Niet van toepassing.

## **Tools**

Niet van toepassing.

### **4.3.2.4 Analyseren Productrisico's**



## **Doe**

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

Meer over "Productrisicoanalyse" in paragraaf 2.6.

#### 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.

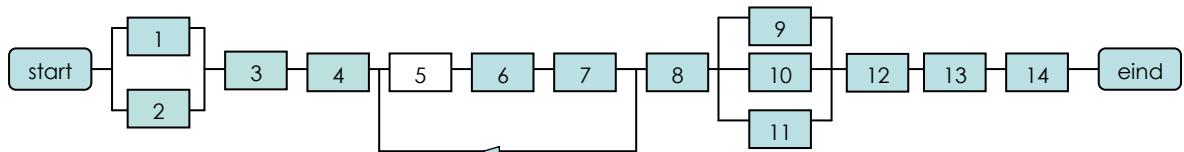
#### Technieken

Productrisicoanalyse (paragraaf 2.6);  
Uitleg kwaliteitsattributen (paragraaf 4.4).

#### Tools

Niet van toepassing.

### 4.3.2.5 Bepalen teststrategie



#### Doeleinden

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 multi-user test, regressietest of ketentest. Op [www.tmap.net](http://www.tmap.net) is een overzicht "Toegepaste testvormen" opgenomen.

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 risicotieverder 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.

#### Testzwaarte

Voor de testzwaarte wordt een keuze gemaakt uit onderstaande mogelijkheden:

- zware test
- gemiddelde test
- beperkte test
- | Implicit 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	ST 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 kenmerk-niveau 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	GAT MTP	Deelsys1	Deelsys2	Totaalsys
Functionaliteit	n.v.t.	n.v.t.	A/●● functionele	B/-	C/● regressive
Gebr.vriendelijkheid	B	●●	C/I	-	B/●● usability
Beveiliging	A	●			
- autorisatiematrix	B		-/S autorisatie-test	-/S autorisatie-test	B/●● procestest
- applicatie	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 AT-omgeving. 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 tussenreleases (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 teveel terugkoppelings-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 is het heel makkelijk om een lichtere of zwaardere regressietest uit te voeren. Dit geeft veel flexibiliteit bij het testen van latere releases.

### 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

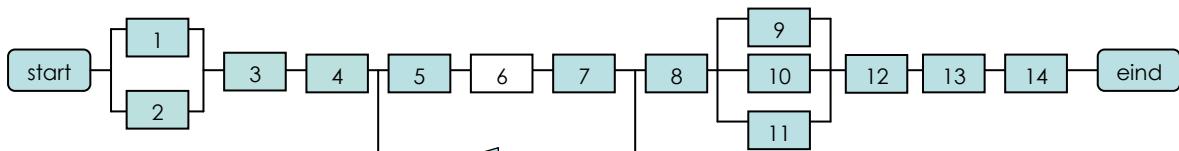
Strategiebepaling (zoals beschreven in deze paragraaf).

### Tools

Niet van toepassing.

#### 4.3.2.6

#### Bepalen begroting



## **Doe**

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 inhoud 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.

## **Uitgediept**

### Begrotingstechnieken

De verschillende begrotingstechnieken en de stappen om tot een begroting te komen staan beschreven in paragraaf 4.9 "Testbegroting opstellen". Voor een testplan kan worden begroot op basis van:

- verhoudingsgetallen
- testobject omvang
- 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 paragraaf 4.10 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 paragraaf 4.10 "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  $\pm 40\%$ , bij start van de testuitvoering wordt dit  $\pm 25\%$ , ergens halverwege wordt het  $\pm 10\%$ .
- De koppeling tussen begroting en diepgang van testen (met bepaalde testtechnieken) is vrij ondoorzichtig. Hoeveel extra tijd kost bijvoorbeeld het toepassen van de elementaire vergelijkingstest 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 er mee 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, beheerde 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.

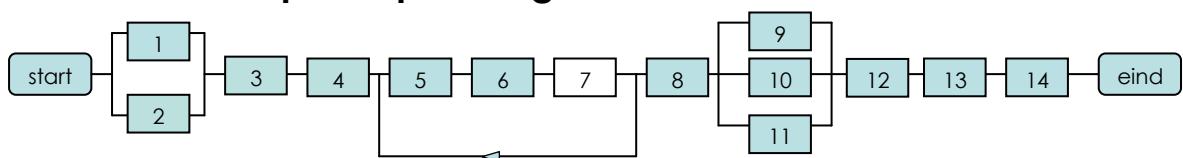
## Technieken

Diverse begrotingstechnieken (paragraaf 4.10);  
Stappenplan voor opstellen begroting (paragraaf 4.10).

## Tools

Plannings- en voortgangsbewakingstools (een begrotingsspreadsheet is te vinden op [www.tmap.net](http://www.tmap.net)).

### 4.3.2.7 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 belangrijk 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 explicet 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 mijlpalmomenten 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	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
PLANNING, BEHEER EN INRICHTING & BEHEER INFRASTRUCTUUR	x														2 FTE
Voorbereiding en Specificatie	x	x	x	x	x	x	x								3480 uren
Uitv. FAT, FIT (functionele integratietest)							x	x	x	x					3480 uren
Uitv. KIT (keten-integratietest)								x	x	x	x				
Uitv. GAT								x	x	x	x				
Afronding												x			
Reserveweek												x			
<b>Totaal:</b>															<b>2 FTE + 6960 uren</b>

##### 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

**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 explicet 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

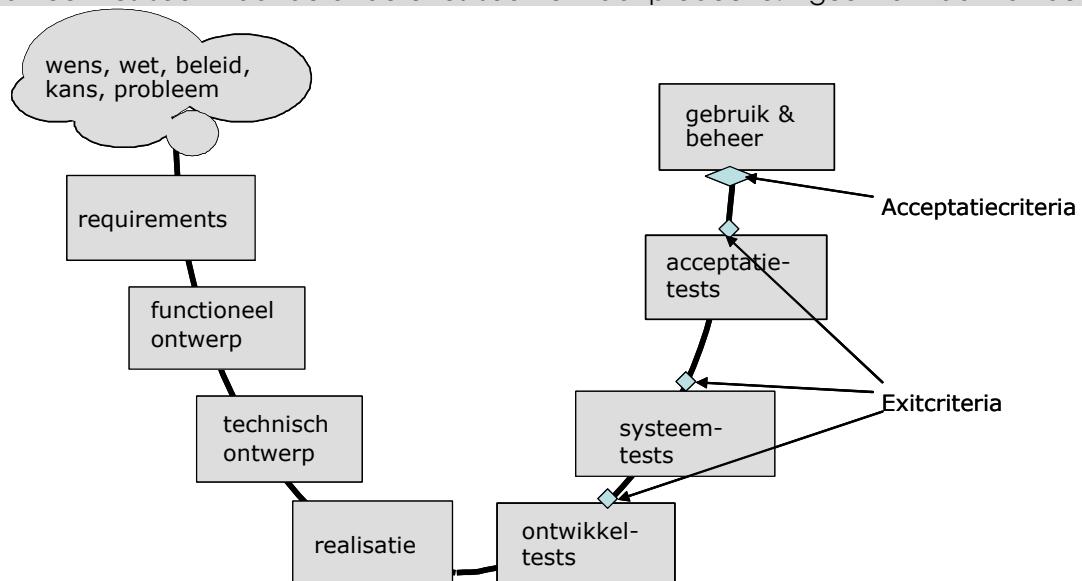
Systeem X mag aan de AT worden overgedragen als schriftelijk aangetoond kan worden dat alle risico's die conform document Y aan de ST zijn toegewezen met de afgesproken diepgang en testwerkwijze zijn getest.

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. 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 43 maakt dit duidelijk.



Figuur 43. 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	ST 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	GAT 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/●● proces- test
- applicatie	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: tolerancies 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.

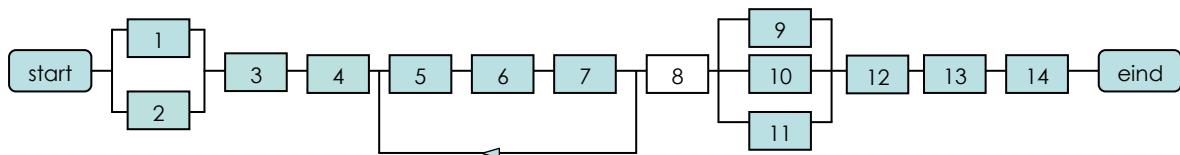
## Technieken

Niet van toepassing.

## Tools

Plannings- en voortgangsbewakingstools.

### 4.3.2.8 Toewijzen testeenheden en testtechnieken



## Doe

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 3 "Website" bevat approaches, dekkingsvormen en 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.
- **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.
- **Arbeidsintensiteit**  
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 meegegenomen 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		Regressietest	te3: selectie uit te1 en te2
Functionaliteit	Deelsys2 (B/●)	Functionele test	te4: Exploratory Testing te5: SYN, SEM
Functionaliteit		Regressietest	te6: selectie uit te4 en te5
Functionaliteit	Totale systeem (C/●)	Integratie	te7: GCT
Functionaliteit		Multi-user	te8: Exploratory Testing
Performance online	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
Gebr.vriendelijkheid	Deelsys1 (C/I)	Gebruikers-vriendelijkheid	impliciet in te1 en te2
Gebr.vriendelijkheid		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. De autorisatiematrix wordt eerst steekproefsgewijs gecontroleerd op juiste vulling, daarna worden de autorisaties met behulp van de semantische testtechniek getest. De lichte penetratietest gebeurt met error guessing, Inpasbaarheid wordt met de procescyclustest getest.

Is de keuze gemaakt om 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](http://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	Penetratie- test
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*	Usability-test (eventueel in lab)
Infrastructuur		RLT*	

Kwaliteitsattribuut	Testontwerptechniek		
	• / licht	•• / gemiddeld	••• / zwaar
(geschiktheid voor)			
Inpasbaarheid	PCT testmaat-1 UCT*	PCT	PCT testmaat-3
Performance		RLT	
Portabiliteit	CKL  Steekproef functionele tests  Steekproef omgevings-combinaties	Functionele regressietest  Belangrijke omgevings-combinaties	Alle functionele tests  Alle omgevings-combinaties
Zuinigheid		RLT	

\* Wanneer de techniek enigszins wordt aangepast, is deze bruikbaar voor het testen van betreffende kwaliteitsattribuut

Toelichting bij de bovenstaande tabel:

Gebruikte afkortingen:

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 paragraaf 3.7: "Een basisset 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  $3 \times 4 = 12$  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 test gevallen.

#### 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.

#### 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.

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.

#### Uitgediept

##### 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.

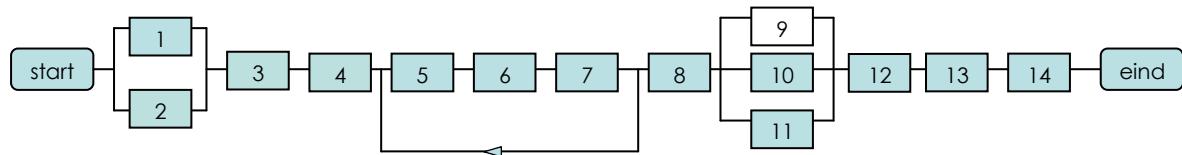
#### **Technieken**

Niet van toepassing.

#### **Tools**

Workflowtool.

### 4.3.2.9 Definiëren testproducten



#### Doele

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 hertests en 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.
- 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 uitgangs databases;
- 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.

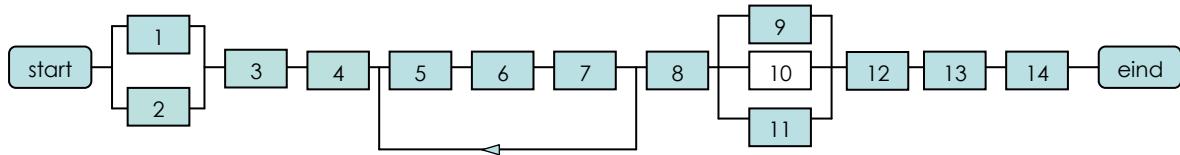
## **Technieken**

Niet van toepassing.

## **Tools**

Testwarebeheertools.

### 4.3.2.10 Definiëren organisatie



#### Doe

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 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.

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 (zie ook paragraaf 2.4 "Testorganisatie")

De samenhang tussen de genoemde rollen en de relaties met de andere betrokken partijen binnen het systeemontwikkelproces 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 (zie figuur 44):

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.

	<b>zelfstandige activiteit</b>	<b>geïntegreerd</b>
<b>project-organisatie</b>	acceptatietest, traditionele systeemtest	ontwikkeltests, systeemtest in agile omgeving
<b>lijn-organisatie</b>	testfabriek	onderhoudsproces

Figuur 44. 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

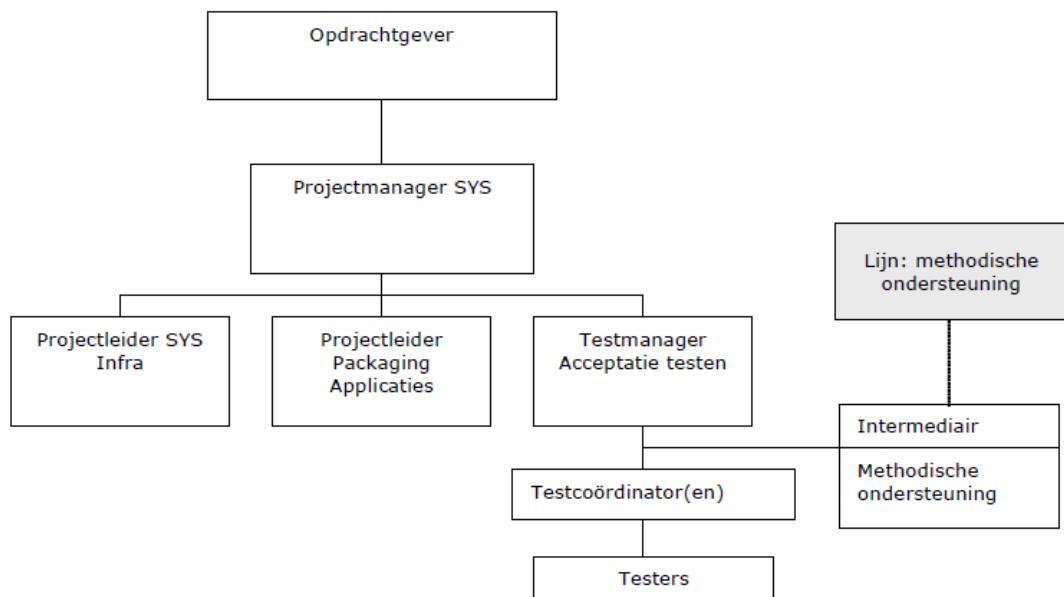
Testen geïntegreerd in lijnorganisatie

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.

Voordelen:

- 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.

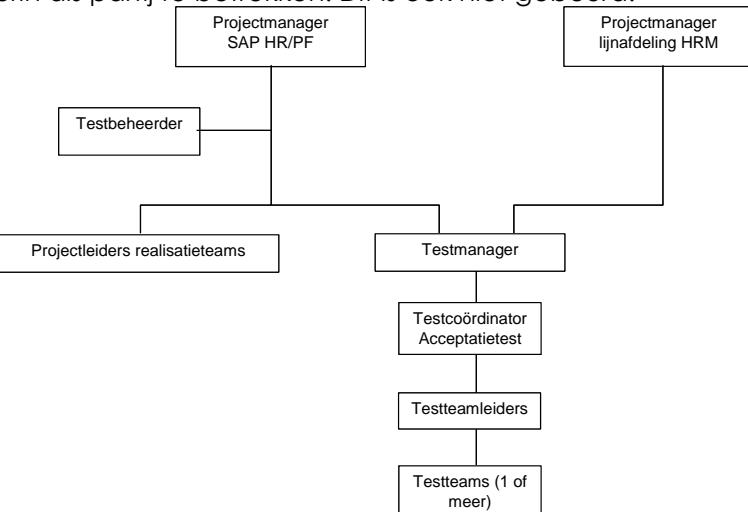


Figuur 45. Voorbeeld traditionele organisatie.

In figuur 45 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.

#### Voorbeeld

In figuur 46 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.

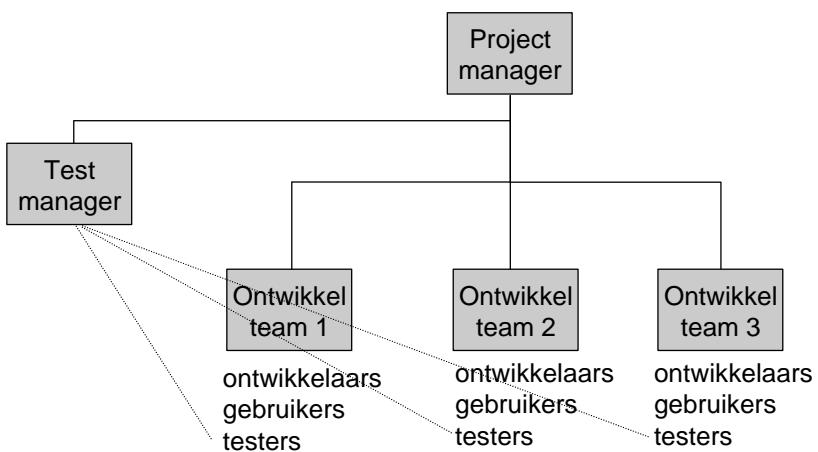


Figuur 46. Voorbeeldorganisatie met twee managers.

#### 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 (zie figuur 47). 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 47. Voorbeeld testmanagers 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 (A), geraadpleegd (C) of geïnformeerd (I) 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.

#### 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. 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](http://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
Groepsworker	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 [Rodden, 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. 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-the-fly" 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
	Opening Vaststellen agenda Mededelingen	xx.xx – xx.xx	<testmanager>
	Notulen vergadering d.d.: <xx-xx-xxxx>	xx.xx – xx.xx	Allen
	Actielijst d.d.: <xx-xx-xxxx>	xx.xx – xx.xx	Allen
	Status, voortgang en kwaliteit: testeenheid 1 testeenheid 2 testeenheid 3 ...	xx.xx – xx.xx	<Tester 1> <Tester 2> <Tester 1> ...
	Kwaliteit testproces <Wat gaat goed en wat kan beter?> <TPI status> <Bevindingenbeheer> <Testwarebeheer> <Reviews> ...	xx.xx – xx.xx	Allen
	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;
  - Gesigneerde (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.

#### 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. 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.

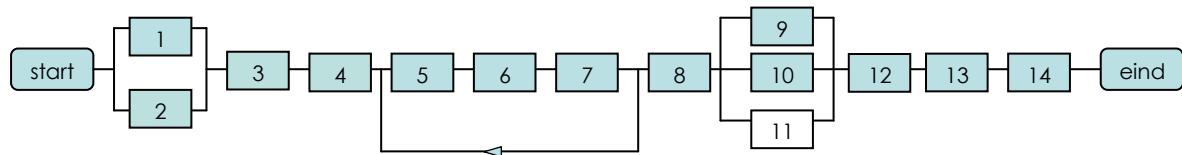
#### **Technieken**

Niet van toepassing.

#### **Tools**

Niet van toepassing.

### 4.3.2.11 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

##### 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](http://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.

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 (werk kamers, vergader kamers, telefoons, PC's, netwerkaansluitingen, kantoor software, 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](http://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.

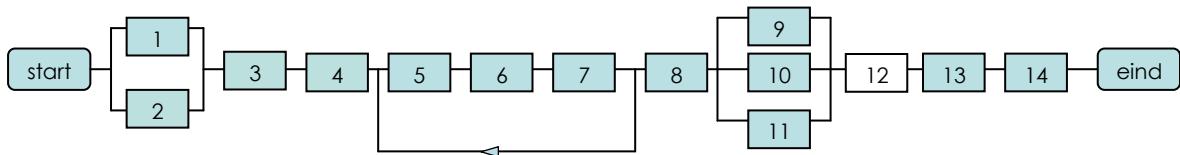
## Technieken

Checklist "Testomgevingen" ([www.tmap.net](http://www.tmap.net));  
Checklist "Kantoorinrichting" ([www.tmap.net](http://www.tmap.net)).

## Tools

Niet van toepassing.

### 4.3.2.12 Inrichten beheer



#### Doele

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. Hier toe 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: testprojectadministrator.

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
<b>User Stories (test basis)</b>				
Totaal	nvt	103	23	nvt
Status 1 (bedacht)	nvt	62	23	nvt
Status >= 2 (opgepakt door	nvt	41	0	nvt

	Regressie	Deelsys1	Deelsys2	Deelsys3
testen)				
<b>Handmatige testscripts</b>				
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
<b>Resultaten laatste handmatige testronde</b>				
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
<b>Geautomatiseerde testscripts</b>				
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
<b>Resultaten laatste geautomatiseerde testronde</b>				
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
<b>Bevindingen</b>				
Nieuw	9	13	nvt	0
Open	197	1	nvt	22
In oplossing	20	5	nvt	0
Te herstellen	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 paragraaf 4.7 "Bevindingen". 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:

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 paragraaf 4.10 "Metrieken".
- 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 systeemontwikkelproces geldende normen en standaards voor documentatie en configuratiebeheer.

Het beheer wordt soms belegd bij een aparte rol: testwarebeheerder.

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.
- 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 bevindingsbeheer 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 paragraaf 4.7 "Bevindingen". Het beheer wordt soms belegd bij een aparte rol: bevindingenbeheerder.

#### **Producten**

Een beschrijving van de diverse beheerprocessen, vastgelegd in het testplan.

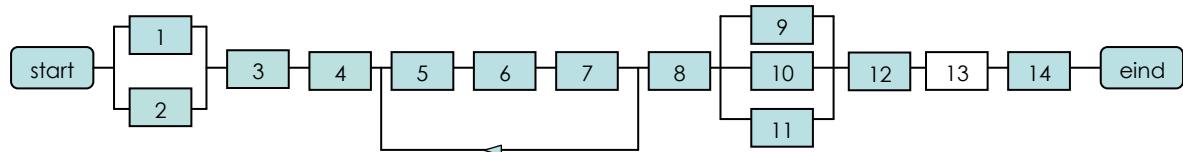
#### **Technieken**

Niet van toepassing.

#### **Tools**

Bevindingenbeheertools;  
Testwarebeheertools;  
Plannings- en voortgangsbewakingstools.  
Workflowtool.

### **4.3.2.13 Bepalen testprocesrisico's (& maatregelen)**



#### **Doe**

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.

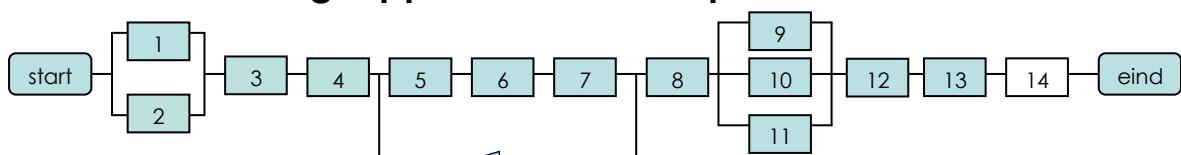
## Technieken

Checklist "Risico's testproces" ([www.tmap.net](http://www.tmap.net)).

## Tools

Niet van toepassing.

### 4.3.2.14 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.

### **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 walkthrough-sessie 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 •. 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.

## **4.3.3. Fase Beheer**

### **Doe**

Het aan de opdrachtgever geven van voldoende inzicht in één 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

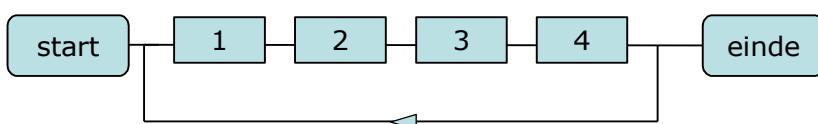
De primair verantwoordelijke rol voor het beheer van het testproces is de testmanager, soms ook testcoördinator genaamd.

## 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 48: Beheer van het testproces

### 4.3.3.1 Uitvoeren beheer

#### Doeleind

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. Het infrastructuurbeheer valt onder de fase "Inrichting en beheer Infrastructuur".
- Het testproces wordt ondersteund met één gecontroleerd op het toepassen van de normen en standaards voor beheer.

Deze werkzaamheden vallen onder de rol van beheerde of testmanager. Hiervoor zijn de volgende beheerde-rollen onderkend: testprojectadministrateur, testwarebeheerde en bevindingenbeheerde.

#### 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 aanwenden 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**

Bevindingenbeheertools;

Testwarebeheertools;

Plannings- en voortgangsbewakingstools.

Workflowtool.

## **4.3.3.2 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.

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.

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 gehertesten 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.

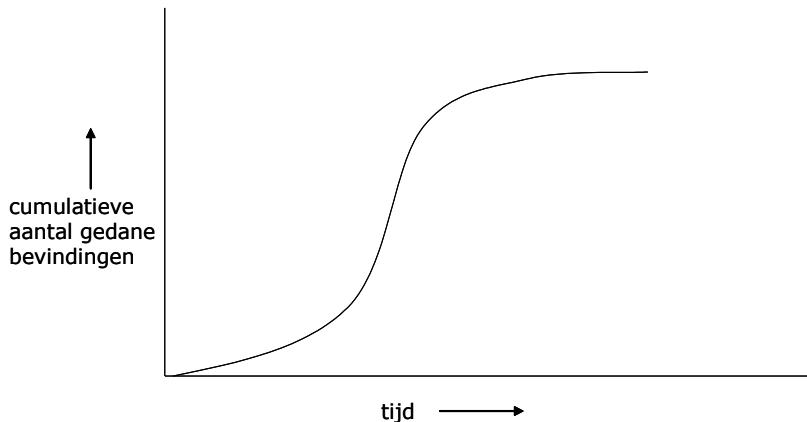
Pas wanneer op deze vragen een positief antwoord gegeven kan worden, is het verantwoord om het advies te geven om te stoppen met testen.

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-productie-name. 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 (zie figuur 49). 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 49. 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, stand-up 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 voordeelig 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	<p>Medewerkers worden vervangen door andere medewerkers die meer ervaring hebben</p> <p>Er wordt extra capaciteit aangevraagd bij de opdrachtgever</p> <p>De testwerkwijze wordt aangepast waardoor minder risicovolle delen nog minder aandacht krijgen en er daardoor meer tijd beschikbaar is voor de risicovolle onderdelen</p> <p>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)</p> <p>...</p>

Gebeurtenis	Het specificeren van testgevallen kost meer tijd dan gepland
Mogelijke oorzaken	<p>De testbasis is van minder kwaliteit dan vooraf ingepland</p> <p>De productiviteit van de medewerkers is lager dan vooraf ingeschat</p> <p>...</p>
Gevolgen voor het testproces	Uitloop van testspecificatie kan betekenen dat niet op tijd met de testuitvoering gestart kan worden
Mogelijke maatregelen	<p>Er worden technieken gekozen die leiden tot minder testgevallen.</p> <p>Dit betekent overigens dat er sprake is van minder testdekking en dat de onderkende risico's minder afgedekt worden</p> <p>De logische testgevallen worden niet uitgeschreven in fysieke testgevallen (dit stelt in het algemeen hogere eisen aan de testuitvoerder)</p> <p>Er wordt extra tijd of capaciteit gevraagd aan de opdrachtgever</p> <p>Extra ondersteuning door materiedeskundigen of ontwikkelaars</p> <p>...</p>

Gebeurtenis	De testbasis verandert steeds
Mogelijke oorzaken	<p>De testbasis is van minder kwaliteit dan vooraf ingepland</p> <p>De scope van het project wordt steeds groter</p> <p>Er zijn meningsverschillen in het project over de te leveren functionaliteit</p> <p>...</p>
Gevolgen voor het testproces	<p>(Tijdens specificatie en uitvoering) De testspecificaties moeten steeds herzien worden en zijn nooit af</p> <p>(Tijdens uitvoering) Er zijn continu extra hertesten nodig</p>
Mogelijke maatregelen	<p>De logische testgevallen worden niet uitgeschreven in fysieke testgevallen (dit stelt in het algemeen hogere eisen aan de testuitvoerder).</p> <p>Exploratory testing als techniek om minder afhankelijk te zijn van de testbasis én deze zo laat mogelijk nodig te hebben</p> <p>Er wordt extra tijd of capaciteit gevraagd aan de opdrachtgever.</p> <p>Strenger configuratie- en changemanagement op projectniveau, (met vanzelfsprekend betrokkenheid van testen)</p> <p>...</p>

Uitgediept  
Hertesten

Een specifiek onderdeel van de strategie is hoe om te gaan met herstellen. Normaal gesproken levert een test bevindingen op die daarna hersteld worden. Voor de herstest moet dan een keuze gemaakt worden. Er kan bijvoorbeeld een beperkte herstest worden uitgevoerd die alleen gericht is op het testen van de aanpassing. Een andere mogelijkheid is om een herstest 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 herstest is gebaseerd op de risico's. Soms zijn er richtlijnen opgesteld, soms bepaalt de testmanager van geval tot geval de gewenste hersteldiepgang. 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 ruimte die de testmanager heeft om zelfstandig maatregelen te nemen wordt bepaald door de volgende factoren:

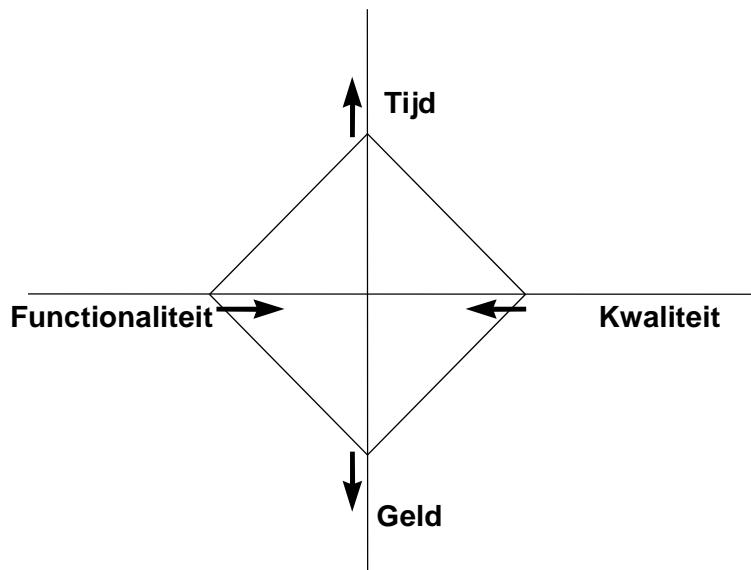
- De mate waarin de problemen voor het testproces kunnen worden opgelost binnen de gestelde opdracht, productrisicoanalyse, teststrategie, begroting, planning en overige randvoorwaarden. Met andere woorden: de opdrachtgever merkt er eigenlijk niets van.
- De mate waarin de problemen kunnen worden opgelost binnen de tolerantiegrenzen, die in de planfase 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 50). 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 50. 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.

### 4.3.3.3 Rapporteren

#### Doe

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 "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)
  - 1.1 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 gaan opleveren;
- Aantal gedane bevindingen en aantal opgeloste (inclusief gehertest) 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 afgedeekt. 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, testsheets, 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.

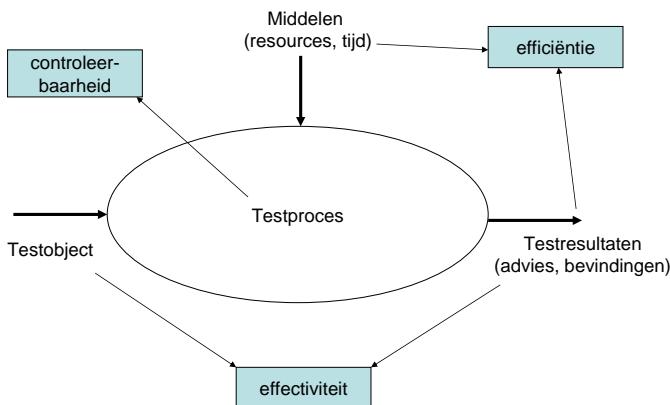
### **5. Afspraken**

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 (zie figuur 51):

- 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 51. 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 paragraaf 4.10 "Metrieken".

### 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 gaan 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 risico-inschatting 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	:(red)	:)	...
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 herstellen	Afgesloten	TOTAAL
<b>Blokkerend</b>			4	4
<b>Ernstig</b>	1	1	12	14
<b>Storend</b>	3	12	49	64
<b>Cosmetisch</b>	15	7	37	59
<b>TOTAAL</b>	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						
<b>TOTAAL</b>	9	20	64	22	7	122

#### Voortgang

Mijlpaal/Activiteit	Tijd			Uren			
	Plan	Verwach	Gerea	Begroot	Bestee	Nog te	Versc

		t	- liseerd	(A)	d (B)	bestede n (C)	hil (A- B-C)
<b>Planning</b>							
- Testplan	1 apr		1 apr	60	54	0	6
<b>Voorbereiding</b>							
- Detailintake	8 apr	12 apr		40	46	4	-10
<b>Specificatie</b>							
- Testeenheid 1	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 schets 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 voórdat 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 'Evaluieren testproces' 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 hertesten.

Een uitgebreide opsomming van de mogelijk te verzamelen ervaringsgegevens is opgenomen in de lijst "Metricslijst". 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 eerder in paragraaf "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 (voortgangsrapport, risicorapport, vrijgaveadvies, eindrapport);

Ervaringgegevens;  
Kosten/batenanalyse.

### **Technieken**

Checklist "Evaluatie testproces" ([www.tmap.net](http://www.tmap.net));  
Lijst "Metricslijst" (paragraaf 4.10).

### **Tools**

Bevindingenbeheertools.  
Testwarebeheertool.  
Plannings- en voortgangsbewakingstools.  
Workflowtool.

## **4.3.3.4 Bijsturen**

### **Doe**

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. Hier toe 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 test gevallen meer op te stellen, maar gebruik te maken van een checklist.
- Veel van de gebeurtenissen die al genoemd zijn bij "Bewaken" 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.

### **4.3.4. 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.

#### 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 testen van een testobject 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 besturingsssoftware, 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.

### 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".

### 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 testvariaties kunnen enorm veel gegevens opleveren. Voorbeeld hiervan is de performancetest waarbij gebruik wordt gemaakt van een testtool. De output van deze testtool kan bestaan 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 en/of mastertestplan. 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

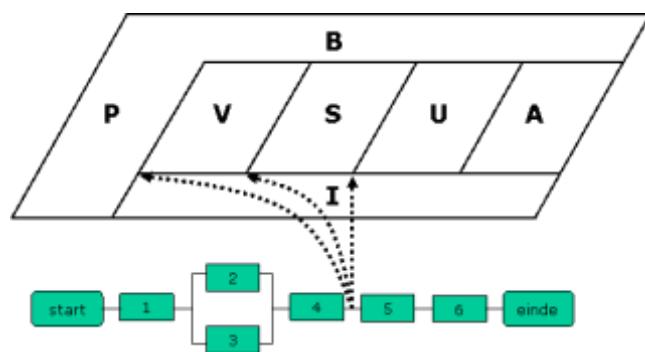
De basis voor de fase Inrichting en beheer infrastructuur wordt gelegd in de fase Planning. Hier wordt in de activiteit "Definiëren infrastructuur" een beschrijving gemaakt van de benodigde infrastructuur op overkoepelend niveau, inclusief planning. Deze beschrijving (uit het mastertestplan of testplan) 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 52 "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 52. 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 52 "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 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.

### 4.3.4.1 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, schijf ruimte, 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 test gevallen.

#### 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.

#### **Producten**

Detailspecificatie werkplek;  
Detailspecificatie testomgeving;  
Plan van aanpak testtool(s).

#### **Technieken**

Niet van toepassing.

#### **Tools**

Niet van toepassing.

### **4.3.4.2 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. 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

### **Producten**

Operationele werkplek;  
Operationele testomgeving;  
Geïnstalleerde testtools.

### **Technieken**

Niet van toepassing.

### **Tools**

Niet van toepassing.

## **4.3.4.3 Specificeren intake infrastructuur**

### **Doeleind**

Het vastleggen van de wijze waarop de intake van de infrastructuur wordt uitgevoerd.

### **Werkwijze**

Doordat de infrastructuur vaak door andere partijen dan het testteam wordtleverd 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.

### 4.3.4.4 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 paragraaf 4.7 "Bevindingen") 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;  
Operationele en bruikbare testomgeving;  
Operationele en bruikbare testtool;  
Intakeverslag.

### **Technieken**

Niet van toepassing.

### **Tools**

Testwarebeheertool;  
Bevindingenbeheertool.

## **4.3.4.5 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.

### **Oplossen knelpunten**

De uitvoering van testscripts kan vertraging opleveren 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 oplevert. 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 is, 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 geoptymaliseerd 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.

### **4.3.4.6      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:

- Selecteren infrastructuur;
- Verzamelen en bijwerken infrastructuur;
- Overdragen infrastructuur.

### **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.

### **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.

#### *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**

Geeconserveerde testinfrastructuur.

### **Technieken**

Niet van toepassing.

### **Tools**

Niet van toepassing.

## **4.3.5. 4.3.5 Fase Voorbereiding**

### **Doeleinden**

Het kunnen beschikken over een, met de opdrachtgever van de test overeengekomen, testbasis die voldoende van kwaliteit is voor het ontwerpen van de test gevallen. 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 (te 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 herstekosten 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.

### Praktijk voorbeeld

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 afgangen.
- 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" 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 hoeft 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.

### 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 testtechnieken 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.

### 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 SMART4 opgesteld) met gebruikmaking van toetstechnieken zoals in paragraaf 4.11 "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".

### Rollen / verantwoordelijkheden

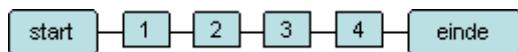
Het opstellen van het rapport detailintake wordt door de testmanager of testcoördinator gedaan. Alle overige activiteiten kunnen door alle testteamleden uitgevoerd. Het rapport is bestemd voor de opdrachtgever van de test.

### 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 53):



Figuur 53. Fase Voorbereiding.

### 4.3.5.1 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

---

4 SMART: S=Specifiek, M=Meetbaar, A=Actueel, R=Realistisch T=Tijdgebonden

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 incrementale 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 geacordeerd 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 oogenschijnlijke 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 foutberichten 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.

## 4.3.5.2 Opstellen checklists

### Doeleind

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](http://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 dekkingsvormen en testontwerptechnieken die in een dergelijke situatie vaak wel toepasbaar zijn:

- Datacombinatietest
- Error guessing
- Exploratory testing
- Grenswaardenanalyse
- Checklist-gebaseerd..

Voor toelichting op en gebruik van deze dekkingsvormen en testontwerptechnieken wordt verwezen naar hoofdstuk 3 "Website".

### Producten

Diverse checklists of één geaggregeerde checklist voor het beoordelen van de testbasis.

### Technieken

Checklist "testontwerptechnieken" ([www.tmap.net](http://www.tmap.net)).

## **Tools**

Niet van toepassing.

### **4.3.5.3 Beoordelen testbasis**

#### **Doe**

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 toetsen technieken en aan de hand van de opgestelde checklist(s) om inzicht te krijgen in de toepasbaarheid van de vastgestelde teststrategie en de daaraan gerelateerde testontwerp technieken. 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 bij activiteit "Inrichten beheer" vastgestelde procedures.

#### **Producten**

Testbasisbevindingen.

#### **Technieken**

Checklist voor beoordelen testbasis  
Toetsen technieken (paragraaf 4.11).

#### **Tools**

Bevindingenbeheertool.

### **4.3.5.4 Opstellen rapport detailintake**

#### **Doe**

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".

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.

## **4.3.6. 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 testgevallen 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**

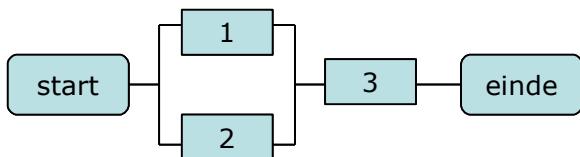
De activiteiten in de fase Specificatie worden uitgevoerd door de testers.

### **Activiteiten**

Binnen de fase Specificatie worden de volgende activiteiten onderkend:

1. Opstellen testspecificaties;
2. Definiëren centrale uitgangssituatie(s);
3. Specificeren intake testobject;

Schema 53 geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten. Activiteiten 1 en 2 beïnvloeden elkaar over en weer.



Figuur 54. Fase Specificatie.

### **4.3.6.1 Opstellen testspecificaties**

#### **Doeleind**

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.

#### 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 wijzigingsvoorstellingen, 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. 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 kan gebruik gemaakt worden van testontwerptechnieken (zie hoofdstuk 3). 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

Meer over het toepassen van deze stappen kan gevonden worden in hoofdstuk 3.

## 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 toetsen 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](http://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.

### **Opbouwen en beheren schaalbare regressietests**

#### **Uitgediept**

In de praktijk zijn regressietests vaak gebrekig 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;
- snell en adequaat te rapporteren over de voortgang van testspecificatie en/of testuitvoering;
- testtrajecten nauwkeurig te plannen en te begroten;
- snell 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 'geklassificeerd'. 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

Exploratory Testing (ET) is 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.

### **Producten**

Bevindingen op de testbasis;  
Testspecificaties (checklists, testgevallen, testscripts).

## **Technieken**

Approaches, dekkingsvormen en testontwerptechnieken (hoofdstuk 3)  
Checklists voor diverse kwaliteitsattributen, [www.tmap.net](http://www.tmap.net)

## **Tools**

Testdatatool;  
Testontwerptool;  
'Model based testing'-tool;  
Testwarebeheertool;  
Geautomatiseerde testuitvoeringstool.

### **4.3.6.2 Definiëren centrale uitgangssituatie(s)**

#### **Doe**

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 maand batches 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 inconsistent 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 test gevallen. 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 test gevallen.

**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 uitgangssituatie/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 0-scripts 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 (wegen 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 kopiëren 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 en 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 uitvoervoorstellingen, 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 test gevallen 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 test gevallen.
- Vul op zijn beurt subset X+1 ook weer aan met test gevallen 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 test gevallen (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 test gevallen in deze systemen.
- Draai de extractie en controleer het resultaat in de datawarehouse.
- Zet als een soort regressietest dezelfde test gevallen nu in de volle database(s) van de aanleverende systemen.
- Draai de extractie en controleer het resultaat (van de test gevallen) in de datawarehouse.

### 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 beïnvloeding 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 test gevallen 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

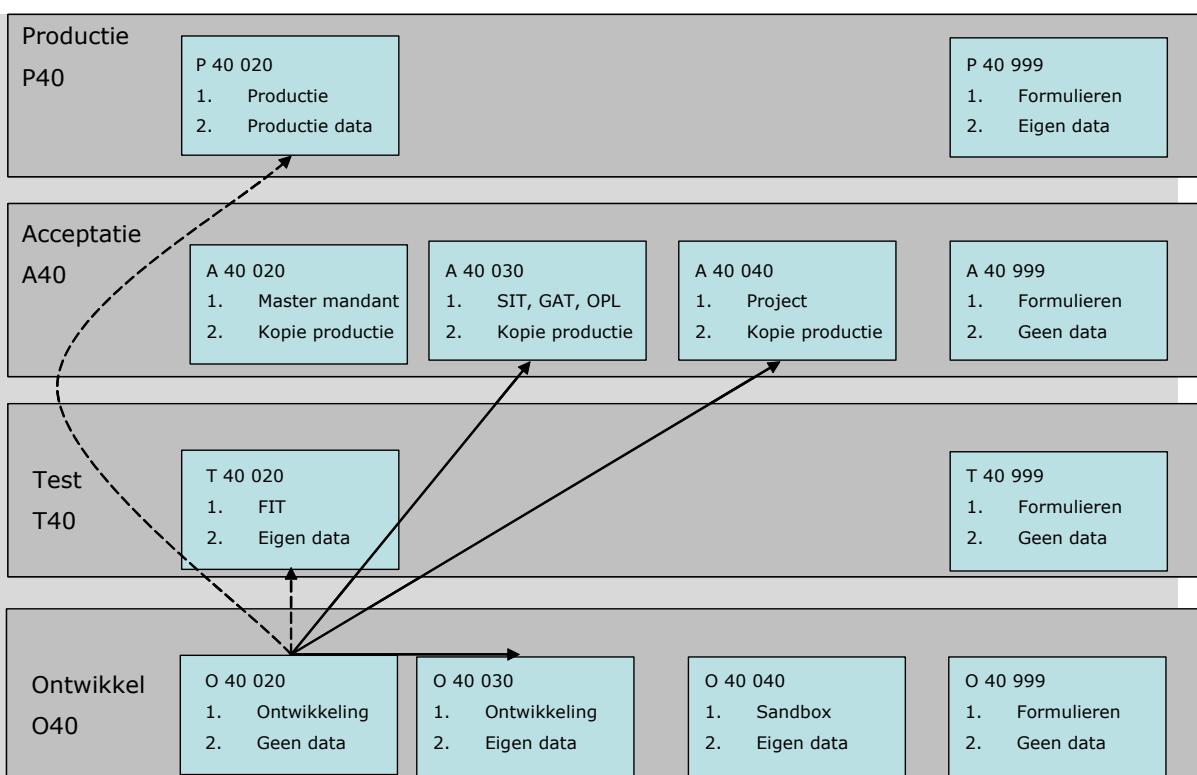
Een derde mogelijkheid is het gebruik van meerdere omgevingen met parallele versies van de gegevens. Elke tester heeft zijn eigen testomgeving en uitgangssituatie(s). Het beschikken over een centrale uitgangssituatie kan handig blijven, maar elke tester kan deze naar believen in eigen omgeving aanpassen. Grote voordeel van deze aanpak is de onafhankelijkheid van de tests: verstoreng 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 isoleren 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 ganonimiseerd overgezet kan worden naar niet productieve omgevingen.



Figuur 55. SAP-omgevingen en transporten.

In figuur 55 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 (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: postcode-tabellen met 3 in plaats van 4 numerieke posities gebruiken of het BTW-percentage 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**

Testdata tool.

### 4.3.6.3 Specificeren intake testobject

**Doe!**

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:

- Opstellen checklist intake testobject;
- Opstellen testscript pretest.

**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.

### 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 test gevallen om te controleren dat de verschillende onderdelen met elkaar kunnen communiceren. De gegevenscyclustest is hiervoor een goede keuze.

De test gevallen 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 test gevallen.
- Bij een andere financiële organisatie duurt de pretest een dag waarin een tester de test gevallen uitvoert, die de belangrijkste functionaliteit bevatten.
- Een telecommunicatie-organisatie laat als pretest een aantal end-to-end test gevallen door de leverancier van de software uitvoeren.

### Producten

Checklist intake testobject;  
Testscript pretest.

### Technieken

Niet van toepassing.

### Tools

Niet van toepassing.

## 4.3.7. 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 geadautomatiseerde) testsheets die in de fase Specificatie zijn opgesteld. In dat geval moeten eerst de uitgangssituaties voor de uit te voeren testsheets 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

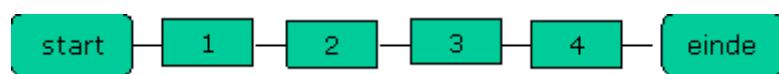
Alle activiteiten kunnen worden uitgevoerd door alle testteamleden. Alleen de controle op volledigheid van het testobject wordt gedaan door de testmanager bijgestaan door (wanneer deze rol is ingevuld) de testinfrastructuurcoördinator.

## 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.

In Schema 55 geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten;



Figuur 56. Uitvoering.

## **4.3.7.1 Intake testobject**

### **DoeI**

Het vaststellen of de opgeleverde delen van het testobject zodanig functioneren dat er zinvol getest kan worden.

### **Werkwijze**

De werkwijze omvat de volgende deelactiviteiten:

- Controle volledigheid opgeleverd testobject;
- Uitvoeren pretest.

#### **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.

#### **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. 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.

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. 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.

### 4.3.7.2 Klaarzetten uitgangssituatie

**Doel**

Het klaar te zetten 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 uitgangssituatie 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. 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 uitgangssituatie en controle daarvan**

Zodra het testobject in de centrale uitgangssituatie 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.

### **4.3.7.3 Uitvoeren (her)tests**

#### **Doe**

Het verkrijgen van testresultaten op basis waarvan de evaluatie van het testobject kan plaatsvinden.

#### **Werkwijze**

De werkwijze omvat de volgende deelactiviteiten:

- Uitvoeren expliciete tests;
- Uitvoeren impliciete tests;
- Uitvoeren toetsen op eindproducten.

## **Uitvoeren expliciete tests**

Bij 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. Expliciet testen is de meest gangbare manier van testen. Er zijn 2 mogelijke vormen van 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 testacties 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 geademtiseerde testuitvoering dan worden de gespecificeerde tests met behulp van een testtool uitgevoerd. Daarnaast kunnen ook tests plaatsvinden op basis van checklists of een andere vorm. Belangrijke voorwaarde voor een waardevolle 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 expliciete test. Dit betekent dat de tester steeds een stukje van de te testen applicatie verkent (exploereert), 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".

### 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, oplegt, 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 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 (gelijkoortige) 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 paragraaf 4.7 "Bevindingen").

#### 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.

#### Uitvoeren impliciete tests

Tijdens het expliciet testen kan ook informatie over andere kenmerken (kwaliteitsattributen) worden verzameld. Hiervoor zijn geen expliciete testgevallen ontworpen. Dit wordt 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 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.

#### Uitvoeren toetsen

In de teststrategie is vastgelegd of er eindproducten moeten worden getoetst. Bij toetsen worden producten beoordeeld zonder dat er sprake is van het runnen van software. Deze toetsing 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 paragraaf 4.7 "Bevindingen").

## **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.

### **4.3.7.4 Controleren en beoordelen testresultaten**

#### **Doe**

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:

- Vergelijken testresultaten;
- Analyseren verschillen;
- Bepalen hertesten.

#### **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

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 systeemdокументatie 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).

### **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 de paragraaf "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 die 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.

### **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 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<sup>5</sup> 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 coverage tool;  
Comparator;  
Databasemanipulatietool.

---

5 Dit wordt ook wel 'daily build' genoemd.

## 4.3.8. 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

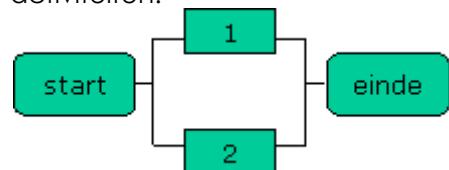
Alle activiteiten kunnen door alle testteamleden uitgevoerd.

### Activiteiten

De fase Afronding bestaat uit de volgende activiteiten:

1. Evaluieren testproces;
2. Conserveren testware.

In schema 56 geeft de volgorde en de afhankelijkheden aan tussen de verschillende activiteiten:



Figuur 57. Afronding.

## 4.3.8.1      **Evaluieren testproces**

### **Doe**

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](http://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) bijeenkomsten. 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](http://www.tmap.net)).

### **Tools**

Testwarebeheertool;  
Bevindingenbeheertool.

## **4.3.8.2 Conserveren testware**

### **Doe**

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:

- Selecteren testware;
- Verzamelen, bijwerken en toegankelijk maken testware;
- Overdragen testware.

Deze activiteit heeft een nauwe relatie met de activiteit “Conserveren infrastructuur” in de fase “Inrichting en beheer infrastructuur” .

Tip

#### **Eerder beginnen met de activiteit “Conserveren Testware”**

Ook al is de activiteit Conserveren Testware de laatste activiteit in het TMap faseringssmodel, 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 hierbij helpen. De manier waarop dit gebruik van standaarden is ingericht, wordt gedefinieerd tijdens de activiteit “Inrichten beheer” in de fase Planning.

### **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.

### **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 aannname 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.

#### **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.

## **4.4 Kwaliteitsattributen**

Voor het testen hanteert TMap de onderstaande set van kwaliteitsattributen. Een andere bekende set van kwaliteitsattributen is te vinden in de internationale standaard ISO25010. Het hanteren van een set kwaliteitsattributen, hetzij van TMap, hetzij ISO25010, 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.

## Uitgediept

Er is een aantal redenen om de TMap set van kwaliteitsattributen te (blijven) hanteren en niet over te stappen op ISO25010:

- 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. ISO25010 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;
- ISO25010 is niet noodzakelijk beter of slechter dan de TMap set, het is alleen anders;
- Hoewel ISO25010 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 ISO25010 als standaard. Ook hanteren diverse organisaties nog oude versies van ISO25010 (bijv. 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.

### **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 worden getoetst door de aanwezigheid van maatregelen en hulpmiddelen die het beheer vereenvoudigen c.q. mogelijk maken, te beoordelen. 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.

## **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.

## **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 getoetst door de betreffende maatregelen (zoals standaardisatie) met behulp van een checklist te beoordelen. Testen van connectiviteit gaat dus over het beoordelen of toetsen 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;
- uitwijkmöglichheid: het gemak waarmee (een deel van) de informatievoorziening op een andere locatie kan worden voortgezet.

Continuïteit kan worden getoetst door de opzet en het bestaan van de maatregelen in het kader van continuïteit te beoordelen aan de hand van een checklist. Implicit 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 (robust, herstelbaar, degradatie en uitwijk) zijn 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, vierkantstellingen en audit-trail. Controleerbaarheid kan worden getoetst

gericht op de opzet van de betreffende maatregelen met behulp van een checklist en kan explicet 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 aangepast. Oftewel: de mate waarin het systeem door de beheerorganisatie kan worden aangepast, zonder afhankelijk te zijn van de automatiseringsafdeling voor onderhoud.

Flexibiliteit wordt getoetst door de betreffende maatregelen met behulp van een checklist te beoordelen. Explicet 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 consistent databases en uitvoer;
- volledigheid: de zekerheid dat alle 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 explicet 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.

### **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 getoetst 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 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 systeemdокументatie) met behulp van een checklist te beoordelen. Inzicht in de gestructureerdheid van de programmatuur (een aspect van onderhoudbaarheid) wordt verkregen door het uitvoeren van toetsactiviteiten, bij voorkeur ondersteund door codeanalysetools .

## **Performance**

De snelheid waarmee het informatiesysteem interactieve en batch-transacties afhandelt.

## **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.

## **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 systeemdокументatie. Dit wordt getoetst 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 systeemdокументatie;
- 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 expliciet getest met behulp van tools die het te onderzoeken middelenbeslag meten en/of impliciet door het opbouwen van statistieken (door

dezelfde tools) tijdens het uitvoeren van tests gericht op functionaliteit. Met name bij embedded systemen speelt dit aspect vaak sterk.

#### 4.4.1.1 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

##### (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 wijzigingsvoorstellingen. 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

Régressie 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 onschabare waarde. Zeker in de beheersituatie biedt de test het vertrouwen dat de nieuwe versie van het systeem of pakket als geheel nog goed werkt.

### **Testontwerptechnieken**

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 nieuwbuwtests 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 nieuwbuw 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 "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.

## **(2) Usability**

### **Wat is usability?**

Zoals voor de meeste IT-definities zijn er ook voor usability diverse definities te vinden.

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](http://www.w3c.org)].

### **Belang van usability testen**

Ontegenzeggelijk is het belang van usability enorm toegenomen met de toenemende digitalisering en informativering 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 usability 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.

### **Testontwerptechnieken**

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 applicatielnavigatie 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](http://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:

- 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.
- 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 gevuld. 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](http://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?

## **4.5 Testomgeving**

### **4.5.1. Inleiding**

Voor het 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 "Testomgevingen toegelicht" wordt gedefinieerd wat een testomgeving is, waarna in paragraaf "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 ("Problemen bij testomgevingen") worden kenmerkende problemen beschreven rondom testomgevingen, gevolgd door een oplossing ter voorkoming van deze problemen: het OTAP-model in paragraaf "OTAP-model".

#### 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 besturingsssoftware, 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 unit-test 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 softwareconfiguratieën 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.

## **4.5.2. Inrichting van testomgevingen**

### **4.5.2.1 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.

### **(1) 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.

### **(2) 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.

### **(3) 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.

### **(4) Continu**

In het geval van verstoorende 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 uitwijkmöglichheid te creëren voor de testomgeving. De uitwijkmöglichheid 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 enigsins beperken van de kosten, kan er voor gekozen worden, deze te combineren met de uitwijkmöglichheid 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 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.

### 4.5.2.2 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.

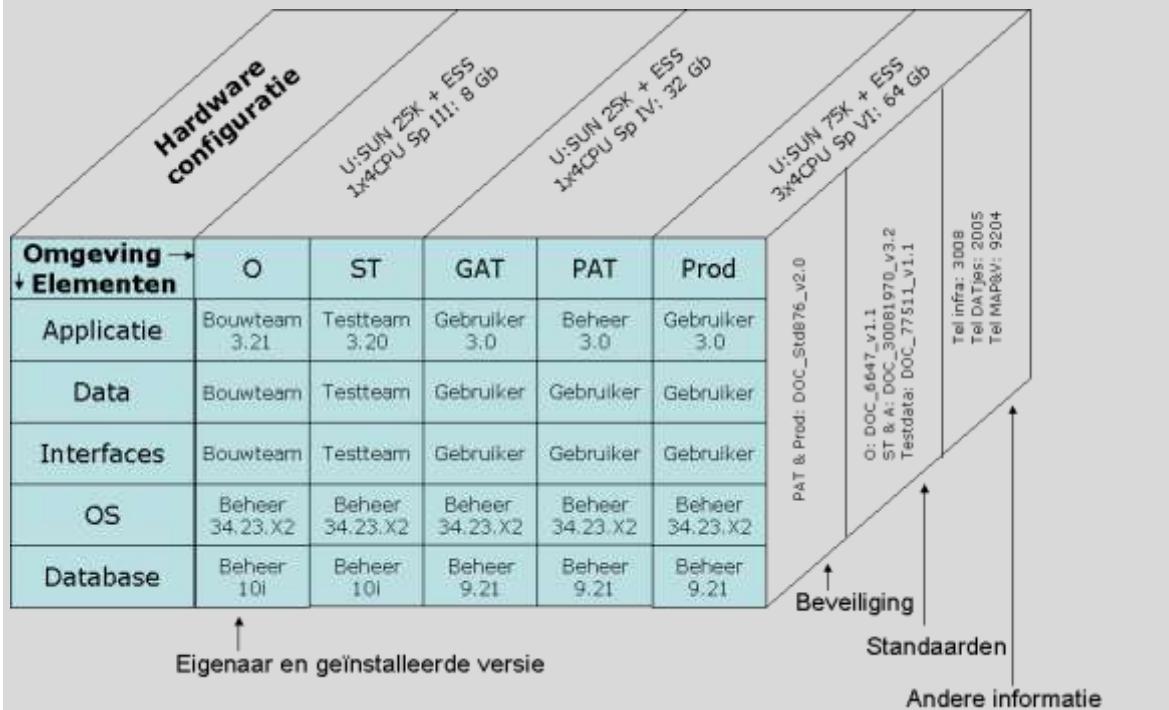
- De testsoort waarvoor de omgeving bedoeld is. Unit-, systeem- of acceptatietest of eventueel een gecombineerde test;
- De testvorm waarvoor de omgeving bedoeld is. Performance-, usability-, security- of regressietest?
- Eisen die aan de omgeving worden gesteld vanuit de externe organisaties. Dit kunnen 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 58 "Kubusnotatie van de verschillende kenmerken van een testomgeving" weergegeven kubus een aantal kenmerken gezet. Hieronder staat een voorbeeld.



Figuur 58. 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.

### **4.5.3. 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 (4.5.4 "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 "Processen bij testomgevingen" aan bod.

### **4.5.4. 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 schrijft 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 59 "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 ontwikkelomgeving	Centrale ontwikkelomgeving	ST omgeving	GAT omgeving	PAT omgeving	Productieomgeving	Schaduwoomgeving

Figuur 59. 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 (zie figuur 60). 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 gebruikersorganisatie).

Bij de omgevingstypen 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

Figuur 60. 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 paralleliteit van de tests en reductie van de herstekosten 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 fouterherstel. 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 vorhanden 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 vorhanden 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 4.3 "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 4.6 "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.

## **4.6 Testtools**

### **4.6.1. 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. 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 "Testtools toegelicht" wordt toegelicht wat een testtool is. In de paragraaf daarna ("Soorten testtools") worden de verschillende soorten testtools beschreven. Paragraaf "Voordelen gebruik testtools" beschrijft de voordelen van het gebruik van testtools. De laatste paragrafen beschrijven hoe testtools op basis van een toolbeleid ingevoerd kunnen worden bij testorganisaties. Hiervoor wordt eerst in paragraaf "Invoeren van testtools met toolbeleid" het begrip toolbeleid geïntroduceerd en het faseringssmodel beschreven. Daarna komen achtereenvolgens de drie fasen aan bod, te weten de fase Initiatie, de fase Realisatie en als laatste de fase Exploitatie.

#### 4.6.2. 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. Structureren 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.

#### 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.

#### **4.6.3. 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:

- 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 vormgeven van de testomgeving

### **4.6.3.1 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.

#### **(1) 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.

#### **(2) 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.

#### **(3) 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](http://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.

## **(4) 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.

### **4.6.3.2 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 dekkingsvorm. 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:

- Testontwerptool;
- 'Model based testing'-tool.

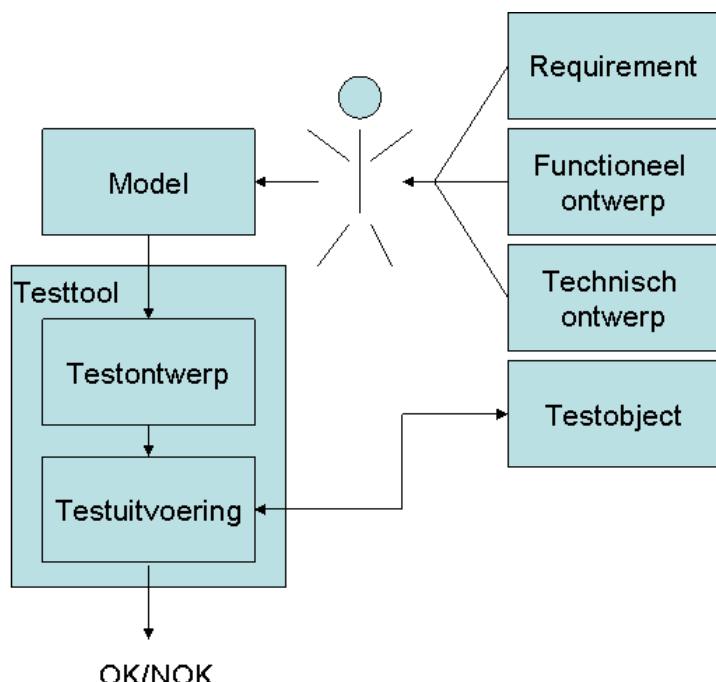
#### **(1) Testontwerptool**

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.

#### **(2) '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 61: 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](http://www.model-based-testing.org).



Figuur 61. 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

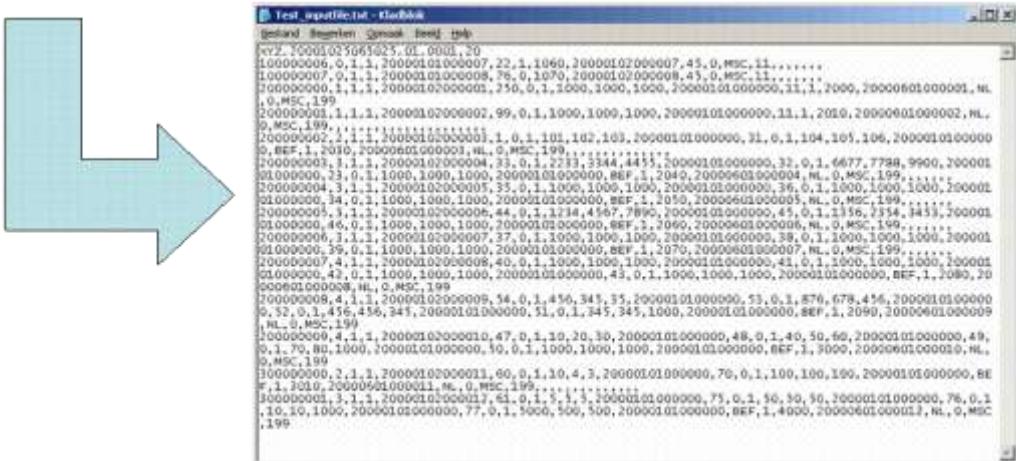
Een batchsysteem maakt als invoer gebruik van tekstbestanden. Deze tekstbestanden bevatten regels met verschillende data. Iedere regel bestaat uit 10 of meer data-elementen en deze worden van elkaar gescheiden door een komma. Het testen van dit batchsysteem gebeurt door middel van deze tekstbestanden. 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 data-element 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 62 "van een spreadsheet voor de creatie van een tekstbestand" voor een schematische weergave.

#### Invoer via spreadsheet:

Nr	J01	J02	J03	Date	Cor	RT	DDF	Last_File	00	XL	MAP	00	R1	R2	R3	R4	R5
100000006	0	1	1	20000101000007	20	1	1000	20000102000007	45	0	MSC	11					
100000007	0	1	1	20000101000008	76	0	1070	20000102000008	45	0	MSC	11					
200000001	1	1	1	20000102000001	250	0	1	1000	1000	1000	20000101000001	11	1	2000	20000001000001	NL	0
200000001	1	1	1	20000102000002	99	0	1	1000	1000	1000	20000101000002	11	1	2010	20000001000002	NL	0
200000002	2	1	1	20000102000002	1	0	1	101	102	103	20000101000000	31	0	1	104	105	106
200000003	3	1	1	20000103000004	33	0	1	223	3344	4456	20000101000000	32	0	1	6877	7778	9500
200000004	3	1	1	20000103000005	35	0	1	1000	1000	1000	20000101000000	36	0	1	1000	1000	1000
200000005	3	1	1	20000103000006	44	0	1	1234	4567	7890	20000101000000	36	0	1	1356	2364	3455
200000006	3	1	1	20000103000007	37	0	1	1000	1000	1000	20000101000000	38	0	1	1000	1000	1000
200000007	4	1	1	20000103000008	40	0	1	1000	1000	1000	20000101000000	41	0	1	1000	1000	1000
200000008	4	1	1	20000103000009	54	0	1	496	345	35	20000101000000	53	0	1	876	679	466
200000009	4	1	1	20000103000010	47	0	1	10	20	30	20000101000000	48	0	1	40	50	60
300000001	4	1	1	20000103000011	60	0	1	10	4	7	20000101000000	70	0	1	100	100	100
300000001	3	1	1	20000103000012	61	0	1	5	5	5	20000101000000	75	0	1	50	50	50

## **Uitvoer in tekstbestand**



Figuur 62. Gebruik van een spreadsheet voor de creatie van een tekstbestand.

#### **4.6.3.3 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.

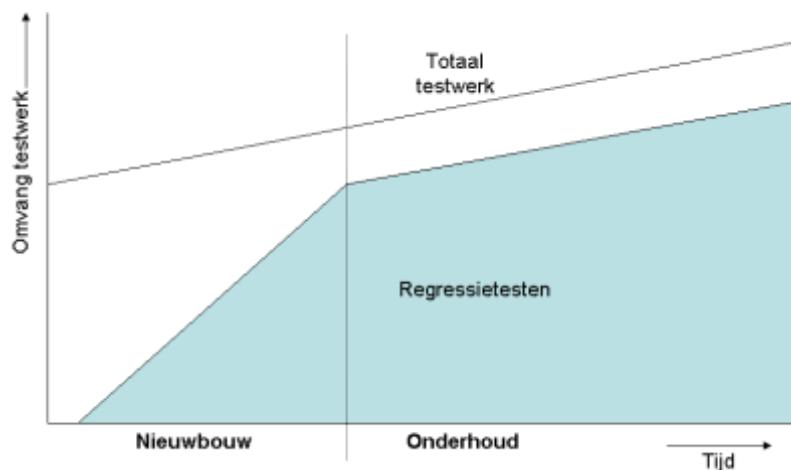
De volgende soorten tools behoren tot deze groep:

- Geautomatiseerde testuitvoering tool;
  - Performance-, load-, en stresstesttool;
  - Monitor;
  - Code coveragetool;
  - Comparator;

- Databasemanipulatietool;

## (1) 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 nieuwembouw van een systeem en krijgt gedurende de verdere levenscyclus een groeiend aandeel in de totale omvang van het testwerk (zie figuur 63 "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 calculatorende testmanager die tientallen procenten wil besparen.



Figuur 63. 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].

## **(2) Performance-, load- en stresstesttool**

Performance-, load- en stresstesttools kunnen een informatiesysteem beladen 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. Om bij de gemeten resultaten aan te kunnen geven waar de mogelijke oorzaken liggen van problemen zijn de tools vaak gecombineerd met monitoren.

## **(3) 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.

## **(4) 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!
- 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.

## **(5) 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<sup>6</sup> of de revisie-functionaliteit van een tekstverwerker te gebruiken.

## **(6) 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 uitgangssituaties. De manipulatietaal waarop dit soort tools zijn gebaseerd is vaak SQL<sup>7</sup>.

### **4.6.4. Tools voor het vormgeven van de testomgeving**

In veel gevallen is een volledig productielijke testomgeving niet zomaar beschikbaar. Er zijn vele mogelijkheden om tools in te zetten om de testomgeving op de juiste manier vorm te geven:

- Simulator
- Stubs en drivers
- Test data tool

## **(1) 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 besturingsssoftware 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.

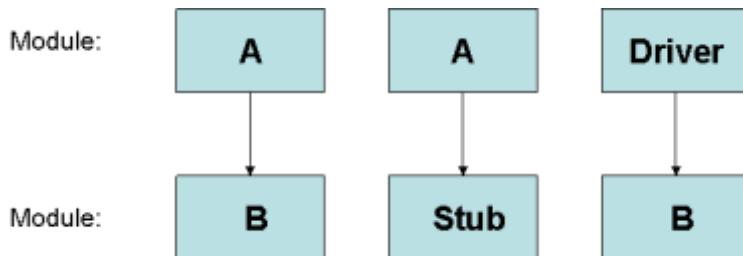
## **(2) Stubs en drivers**

---

6 Wordt vaak standaard meegeleverd met besturingssysteem.

7 Structured Query Language.

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 64 "stubs en drivers in relatie tot module A en module B").



Figuur 64. 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.

### (3) 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).

#### 4.6.5. 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. 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 testdoel 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 lopen. 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.

### **4.6.6. Gewenste effecten, commitment, voorwaarden.**

De eerste fase binnen het faseringssmodel is de fase Initiatie. Binnen deze fase worden activiteiten onderkend die tot doel hebben een eenduidig beeld te krijgen 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 afgenoemt met de belangrijkste belanghebbenden bij het testproces.

Voorbeelden zijn:

- Lijnmanager (verantwoordelijk voor financiën);
- Projectmanager;
- Testmanager;
- Testconsultant;
- Applicatiodeskundige;
- 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 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.

#### **4.6.7. Implementatiefase**

De tweede fase binnen het model is de fase Implementatiefase. 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.

#### **4.6.7.1 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 explicet 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.

#### **4.6.7.2 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 situatie specifiek. Toch zijn er een aantal algemeen geldende voorwaarden te identificeren:

- Testtoolselectie;
- Pilot;
- Business case;
- Management commitment;
- Onderhoud en beheer in de lijn;
- Opleide testers;
- Gestruktueerd testproces;
- Communicatie.

Deze voorwaarden worden hierna verder uitgediept.

## (1) 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](http://www.tmap.net) onder de naam "selectiecriteria testtools".

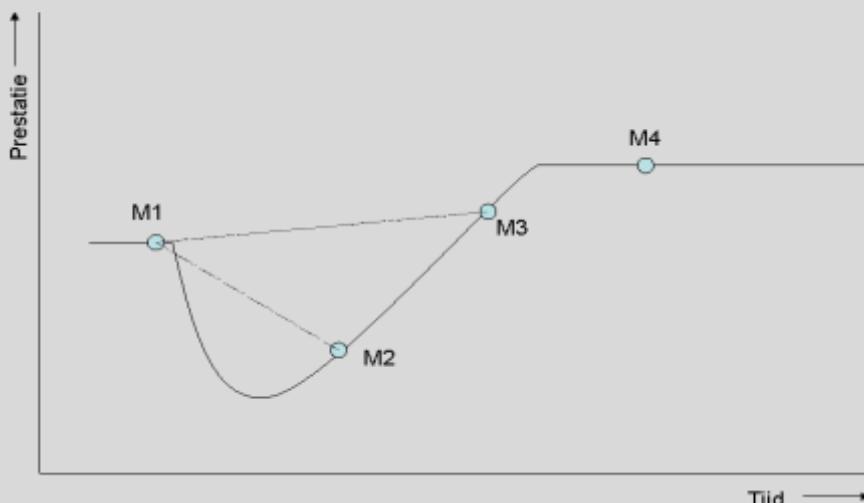
## (2) Pilot

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 65 "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 65. Prestatiedip bij de introductie nieuwe werkwijze.

In de figuur 65 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.

### **(3) 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.

### **(4) 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 tegenval 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 tijd nood komt, is de kans groot dat het gebruik van de tool wordt stopgezet.

### **(5) 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.

## **(6) 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.

## **(7) Gestructureerd testproces**

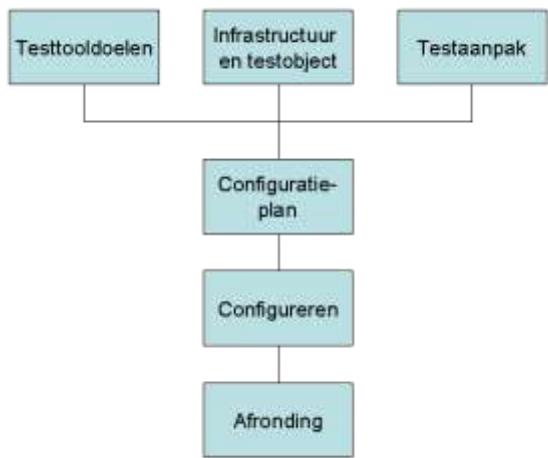
De inzet van testtools richt zich op het verbeteren van het testproces in termen van geld, tijd en kwaliteit. Het levert daarmee een bijdrage aan de verhoging van de efficiëntie van het testproces. Alvorens een testproces efficiënter kan worden ingericht, moet het beheerst worden uitgevoerd. Het kan 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 3 "Website") 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).

## **(8) 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.

## **(9) 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 66 "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 geconfigureerd. 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 66. Configuratie van de testtool.

#### 4.6.8. Gebruik

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 testoaldoelen 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 beheert 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 behevormen 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.

## 4.7 Bevindingen

### 4.7.1. 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 bijklink 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 testinfrastructuur, 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.

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.

#### **4.7.2.      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.

##### **4.7.2.1      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.

#### **4.7.2.2 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 angeven 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.

#### **4.7.2.3 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 collega-tester.

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.

#### **4.7.2.4 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. 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 paragraaf 4.11 "Toetstechnieken".

#### **4.7.2.5 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.

#### **4.7.2.6 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.

#### **4.7.2.7 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 verwiesen 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 verstören. 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.

### 4.7.2.8 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.

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 hierdoor 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.

### **(1) 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.

Voor meer informatie over het doen van een bevinding zie [Black 2004].

### **4.7.3. 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. 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".

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 foute oorzaak 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.

#### **4.7.3.1 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 tracing 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 communicerbaar.
- *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. Normal 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 aangegeven. 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 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.

#### 4.7.3.2 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

- *Hertest*  
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.

#### **4.7.4. 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

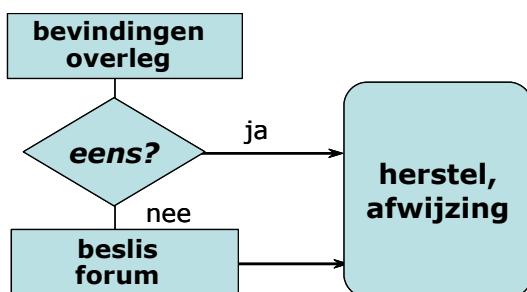
- Conference call  
Wanneer de partijen over verschillende locaties (over de wereld) verspreid zijn, is dit geen reden om geen bevindingenoverleg te voeren. Conference calls of video conferencing bieden dan uitkomst.
- Zorg dat iedere deelnemer goed op de hoogte is van de werkwijze van de bevindingenprocedure 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 67 geeft de relatie tussen bevindingenoverleg en beslisforum weer:



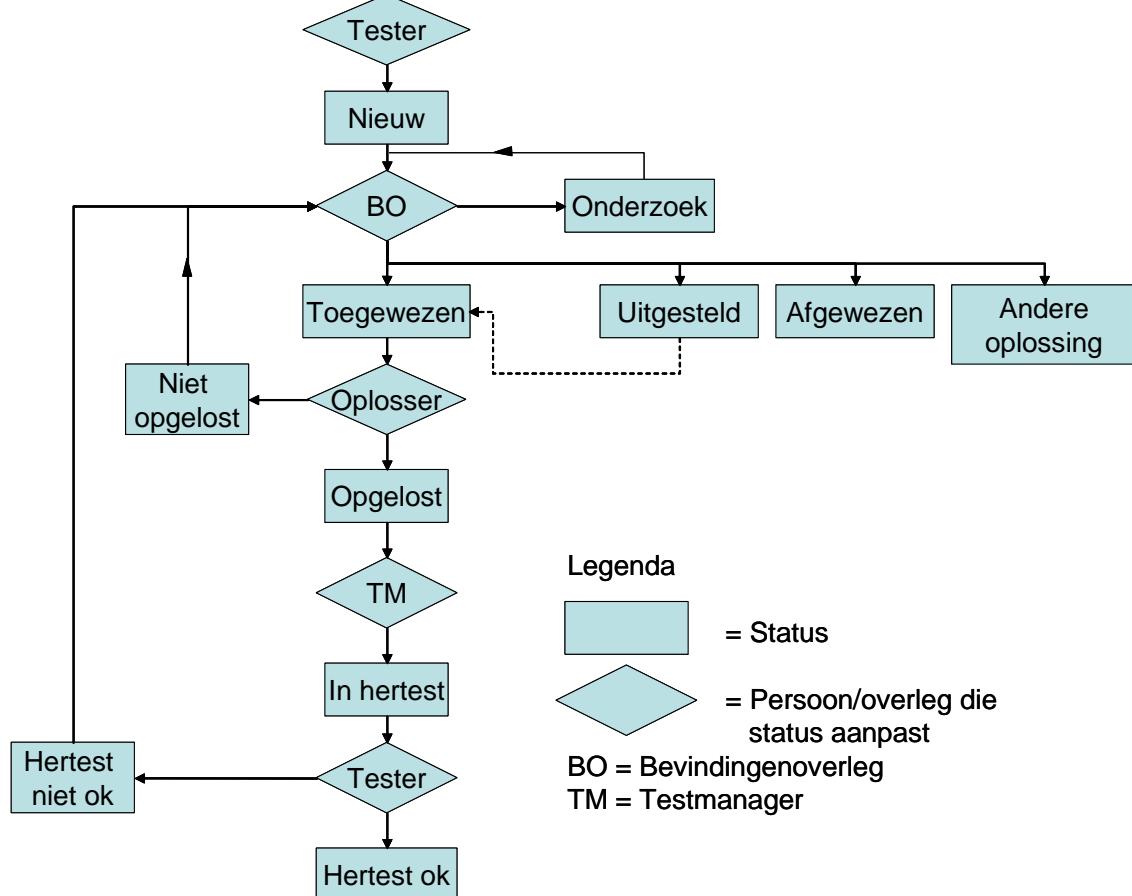
Figuur 67. Bevindingenprocedure.

De oplosser onderzoekt de bevinding en lost deze op. Hier kan ook nog blijken dat de bevinding ontrecht 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.

Figuur 68 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 68. Levenscyclus van een bevinding.

## 4.8 Ontwikkeltesten

### 4.8.1. 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-productie-name te herstellen. De hieruit voortvloeiende langdurige en kostbare herstel- en hertest-cycli zijn voor de meeste organisaties onacceptabel geworden.

Dit hoofdstuk licht in 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 dit hoofdstuk de activiteiten van het ontwikkeltesten volgens het TMap-faseringssmodel.

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 hoe 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.

#### **4.8.2.       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

##### **4.8.2.1      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.

### 4.8.2.2 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 ontwikkeltests 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 voordat 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 ontwikkeltests 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

Figuur 69 [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

Figuur 69. Kenmerken van testers en ontwikkelaars.

#### 4.8.2.3 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).

##### (1) 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?

## (2) Argumenten vóór

Het belangrijkste argument voor meer structuur en diepgang in het ontwikkelttesten 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 ontwikkelttests te detecteren. Wanneer het ontwikkelttesten 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 ontwikkelttests 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 ontwikkelttesten 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 ontwikkelttests. Elke ontwikkelttest in een individueel project moet dan minimaal aan deze eisen voldoen.

### 4.8.2.4 Context van ontwikkelttesten

Het ontwikkelttesten heeft een zeer nauwe relatie met het ontwikkelproces en kan eigenlijk niet los daarvan gezien worden. Ook is voor ontwikkelttesten veel meer kennis nodig van de technische implementatie van het systeem of pakket dan voor een systeem- of acceptatietest. Om het ontwikkelttesten 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 ontwikkelttesten 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.

#### **4.8.2.5 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 ontwikkelteten, 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 ontwikkelteten belangrijkste practices zijn: Pair Programming , Test-Driven Development en Continuous Integration.
- 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 ontwikkelters 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.

#### **4.8.2.6 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.

Testen is 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 voordat deze geïntegreerd worden.

### 4.8.3. **Unittest**

Bij unittesten is het belangrijk om te realiseren wat de plaats van het testen binnen het ontwikkelen is. De unitesters zijn meestal de ontwikkelaars die hun eigen unit testen. De projectleider ontwikkeling, een aparte testcoördinator of de 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 unitests 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 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 unitests worden gegroepeerd in een zogenaamd testraamwerk.

#### Definitie

Een *testraamwerk* (test harness) is een voor een ontwikkelomgeving geconfigureerde verzameling software en testgegevens om één unit of serie van units te testen waarbij het gedrag en de output wordt gecontroleerd.

Het schrijven van unitests 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 maken het testen tot een steeds belangrijker onderdeel van de systeemontwikkeling.

#### 4.8.4. 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.

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 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 geïntroduceerd, die mogelijke problemen bij het combineren van units zover mogelijk naar voren trekt.

## 4.9 Testbegroting opstellen

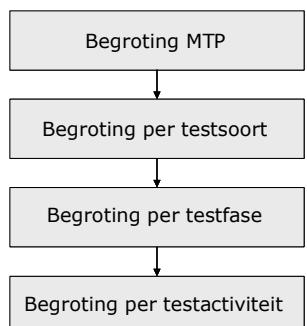
### 4.9.1. Begrotingstechnieken

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

### 4.9.2. Begroten

Er zijn een aantal niveaus waarop een begroting kan worden gemaakt. De verschillende niveaus van begroting staan in figuur 70.



Figuur 70. 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 (unit-test 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 paragraaf 4.10 "Metrieken".

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 tests een onderscheid gemaakt in drie verschillende diepgangen, namelijk •, •• en ••• (laag, gemiddeld en hoog).

	Toets	UT	UIT	Impli-ciete test	Testen	Testen	Testen
<b>Diepgang ⇒</b>	•	•	•		•	••	•••
<b>Kwaliteitsattribuut ↓</b>		2)	2)	3)			
Beheerbaarheid	Tpa-s <sup>4)</sup>				Wbs <sup>5)</sup>	-	-
Beveiliging	Tpa-s				Tpa	Tpa	Wbs
Bruikbaarheid	Tpa-s				Tpa <sup>6)</sup>	Tpa <sup>6)</sup>	Wbs
Connectiviteit	Tpa-s				-	-	-
Continuïteit	Tpa-s				Timebox <sup>7)</sup> week	Timebox <sup>7)</sup> maand	Timebox <sup>7)</sup> kwartaal
Controleerbaarheid	Tpa-s				-	-	-
Flexibiliteit	Tpa-s				-	-	-
Functionaliteit	Tpa-s	Uren-box <sup>7)</sup>	Uren-box <sup>7)</sup>		Tpa	Tpa	Tpa
Gebruikers-vriendelijkheid	Tpa-s				Wbs	Wbs	Wbs
Herbruikbaarheid	Tpa-s				-	-	-
Infrastructuur	Tpa-s				-	-	-
Inpasbaarheid	Tpa-s				Tpa <sup>6)</sup>	Tpa <sup>6)</sup>	Tpa
Onderhoudbaarheid	Tpa-s				-	-	-
Performance					Tpa	Tpa	Tpa
Batch	Tpa-s				Wbs	Wbs	Wbs
Online							
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 bouwinstelling (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 testen van een ander kwaliteitsattribuut. In TPA geeft dat een additionele toevoeging van 0,02 bij het bepalen van de Qd.
  - 4) TPA-s is de toets component van TPA.
  - 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.

#### **4.9.3. 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.

#### **4.9.4. Begroten op basis van testobject omvang**

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 functiepuntdanalyse worden uitgevoerd. Het resultaat van de functiepuntdanalyse is een aantal functiepunten (FP). Eén functiepunkt 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's) 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's	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 vorhanden 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 migratiетraject of een onderhoudstraject kunnen de onderstaande afleidingen wel bruikbaar zijn:

- 1 kilo lines of code (3 GL) = 17 TOM [Collard, 1999].
- [Capers 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

#### **4.9.5. 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 Installeerbaarheid		zacht	24	
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)
Gebruikersvriendelijkheid	••	hard	80	Onderzoeksverzoek grootte (grens: enkele onderwerpen)
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.

#### **4.9.6. 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 toetsen
- de gewenste diepgang
- de formaliteit van de toetsvorm; hoe formeler de toets hoe meer tijd de toetsing kost.

#### **4.9.7. 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)

#### **4.9.8. 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.

#### **4.9.9. 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 functiepunktanalyse (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 functiepunktanalyse 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.

##### **4.9.9.1 Filosofie**

Bij het bepalen van een testbegroting in het kader van een acceptatie- of systeemtest speelt een drietal elementen een rol (zie figuur 72 "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 72. Basiselementen begroten.

#### **4.9.9.2 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 functiepuntenalyse 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?

#### **4.9.9.3 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 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 (risicotrasse 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% risicotrasse A
- 50% risicotrasse B
- 25% risicotrasse C.

Deze verdeling dient vervolgens als uitgangspunt voor het uitvoeren van een TPA.

### 4.9.9.4 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 functiepunktanalyse de relatie tussen inspanningsuren en het gemeten aantal functiepunten. Voor testpunktanalyse betekent productiviteit de tijd die nodig is om één testpunkt, 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.

### 4.9.9.5 Globale werking

Op basis van het aantal functiepunten per functie, de functie afhankelijke factoren (complexiteit, doorwerking, uniformiteit, gebruikersbelang en gebruikssintensiteit) en de kwaliteitseisen c.q. teststrategie met betrekking tot de te meten kwaliteitsattributen wordt per functie het aantal testpunten berekend dat nodig is om de meetbare kwaliteitsattributen te testen (door testen 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 testpunten.

Op basis van het totale aantal functiepunten van het informatiesysteem en de kwaliteitseisen c.q. teststrategie met betrekking tot de toetsbare kwaliteitsattributen, wordt het aantal testpunten bepaald dat nodig is om de door toetsen meetbare kwaliteitsattributen te toetsen (toetsen: testen door het controleren en onderzoeken van producten, zonder dat er sprake is van het uitvoeren van programma's). Dit levert het aantal testpunten voor de toetsactiviteiten – toetspunten dus eigenlijk. Het totaal aantal testpunten ontstaat door de test en de toetspunten 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.

#### 4.9.9.6 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 vorhanden 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<sup>8</sup> 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 functionaliteit en het

8 COSMIC: Common Software Measurement International Consortium

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  $250 \leq CFFP \leq 1000$  :  $FP = CFFP / 1,2$
- bij  $CFFP > 1000$  :  $FP = CFFP / 1,5$

#### TPA, de techniek in detail

#### 4.9.10. 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 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 ( $FP_f$ )

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.

<b>FP<sub>f</sub></b> Registreren	18 functiepunten
<b>FP<sub>f</sub></b> Verwerken	12 functiepunten

(**FP<sub>f</sub>** = functiepunten per functie)

#### 4.9.11. Punten voor testactiviteiten

Het aantal testpunten is een optelling van het aantal testpunten per functie met betrekking tot testbare 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 te testen kwaliteitsattributen ( $Q_d$ ).

Als eenheid van functie wordt de FPA-functie gehanteerd. Bij het bepalen van het gebruikersbelang en gebruikssintensiteit staat de gebruikersfunctie als communicatiemiddel centraal. Het belang dat door de gebruikers wordt toegekend aan de gebruikersfunctie geldt tevens voor alle onderliggende FPA-functies.

##### 4.9.11.1 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.

##### 4.9.11.2 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.

## **(1) Wegin**

- 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.

### **4.9.11.3 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 gebruiksinintensiteit toegekend aan door gebruikers benoemde functionaliteit c.q. de gebruikersfunctie.

## **(1) Wegin**

- 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.

### **4.9.11.4 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's	Functies		
	1	2 - 5	> 5
1	L	L	G
2 - 5	L	G	H
> 5	G	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.

## **(1) Wegin**

- 2 de functie heeft een lage doorwerking
- 4 de functie heeft een gemiddelde doorwerking

8 de functie heeft een hoge doorwerking.

#### 4.9.11.5 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 IF-statements 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.

#### (1) Wegining

- 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.

#### 4.9.11.6 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 functiepuntnalyse als uniek worden aangemerkt. Binnen de functiepuntnalyse 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 functiepuntnalyse 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.

#### 4.9.11.7 Berekeningswijze

De factor ( $A_f$ ) wordt bepaald door de som van de waarden van de eerste vier functieafhankelijke variabelen (gebruikersbelang, gebruikssintensiteit, 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.

$$A_f = ((G_b + G_i + S_d + C) / 20) * U$$

$A_f$  = wegingsfactor van de functie-afhankelijke factoren

$G_b$  = gebruikersbelang

$G_i$  = gebruikssintensiteit

$S_d$  = systeemdoorwerking

$C$  = complexiteit

$U$  = uniformiteit

#### 4.9.11.8 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	$G_b$	$G_i$	$S_d$	$C$	$U$	$A_f$
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 ( $A_f$ )

	<u>Registreren</u>	<u>Verwerken</u>
Gebruikersbelang	6	12
Gebruiksintensiteit	8	2
Systeemdoorwerking	2	2
Complexiteit	3	6
Uniformiteit	1	1
$A_f =$	$19/20 * 1 = 0,95$	$22/20 * 1 = 1,10$

(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.)

#### 4.9.11.9 Testbare kwaliteitsattributen

Hierna wordt aangegeven hoe de eisen die worden gesteld ten aanzien van de testbare kwaliteitsattributen worden meegenomen binnen de testpuntanalyse. Met betrekking tot de testbare kwaliteitsattributen wordt in TPA onderscheid gemaakt tussen kwaliteitsattributen die expliciet en/of impliciet kunnen worden gemeten.

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.

### **(1) Wegining**

- 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 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 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 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 impliciet worden getest.

#### **4.9.11.10 Berekeningswijze ( $Q_d$ )**

Per 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 expliciet meetbare kwaliteitsattributen, worden opgeteld.

Indien er voor gekozen is bepaalde kwaliteitsattributen 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 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 testbare kwaliteitsattributen ( $Q_d$ )**

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	$(0/4) * 0,05 = 0$

Implicit worden gemeten:  
 Performance = 0,02  
 Zuinigheid = 0,02  
 Onderhoudbaarheid = 0,02

$$Q_d = 0,94 + 0,05 + (3 * 0,02) = 1,05$$

#### **Formule testpunten voor testactiviteiten**

Het aantal 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:

$$TP_f = FP_f * A_f * Q_d$$

TP<sub>f</sub> = het aantal testpunten per functie

FP<sub>f</sub> = aantal functiepunten per functie

A<sub>f</sub> = wegingsfactor van de functie-afhankelijke factoren

Q<sub>d</sub> = wegingsfactor van de testbare kwaliteitsattributen

#### **Rekenvoorbeeld (4): Berekening testpunten (TP<sub>f</sub>)**

	FP <sub>f</sub>	*	A <sub>f</sub>	*	Q <sub>d</sub>	=	TP <sub>f</sub>
Registreren	18		0,95		1,05	=	18
Verwerken	12		1,10		1,05	=	<u>14 +</u>
Totaal aantal testpunten							32

#### **4.9.12. Toetsbare testpunten**

Het aantal toetsbare testpunten is met name afhankelijk van de door toetsen te testen kwaliteitsattributen (de factor Q<sub>s</sub>). Daarnaast wordt het aantal toetsbare testpunten beïnvloed door het totaal aantal functiepunten van het systeem. Een beoordeling door toetsen 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 door toetsen 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 worden getest. Bijvoorbeeld beveiliging kan dus met behulp van een semantische test, en of door toetsen, door het beoordelen van de beveiligingsmaatregelen op basis van een checklist, worden gemeten.

#### **4.9.12.1 Berekeningswijze (Q<sub>s</sub>)**

Indien ervoor wordt gekozen een bepaald kwaliteitsattribuut mee te nemen bij de test krijgt de factor Q<sub>s</sub> de waarde 16. Voor elk volgende door toetsen te meten kwaliteitsattribuut die wordt meegenomen wordt de waarde van Q<sub>s</sub> met 16 verhoogd.

#### **Rekenvoorbeeld (5): Berekening toetsbare testpunten (Q<sub>s</sub>)**

Door toetsen (m.b.v. een checklist) worden gemeten:

Continuïteit = 16

Q<sub>s</sub> = 16

### **4.9.13. 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:

$$TP = \sum TP_f + ((FP * Q_s) / 500)$$

TP = het aantal testpunten van het totale systeem

$\sum TP_f$  = de som van het aantal testpunten per functie ( testpunten)

FP = aantal functiepunten van het totale systeem (minimale waarde 500)

Q<sub>s</sub> = wegingsfactor van de toetsbare kwaliteitsattributen

#### **Rekenvoorbeeld (6): Berekening totaal aantal testpunten (TP)**

$$TP = 32 + ((500 * 16) / 500) = 48$$

### **4.9.14. 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.

#### **4.9.14.1 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.

$$V = 1,2$$

## **4.9.14.2 Omgevingsfactor**

Het aantal benodigde testuren per testpunt wordt niet alleen beïnvloed door de vaardighedsfactor, 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.

## **4.9.14.3 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.

### **(1) Wegining**

- 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.

## **4.9.14.4 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.

### **(1) Wegining**

- 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.

## **4.9.14.5 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.

### **(1) Wegining**

- 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.

#### **4.9.14.6      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.

##### **(1)    Wegin**

- 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.

#### **4.9.14.7      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.

##### **(1)    Wegin**

- 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.

#### **4.9.14.8      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.

##### **(1)    Wegin**

- 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.

#### **4.9.14.9      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.

<b>Rekenvoorbeeld (8): Omgevingsfactor (<math>O</math>)</b>
---

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)

$$PT = 48 * 1,2 * 0,95 = 54,72 \text{ (55 uur)}$$

### 4.9.15. 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.

#### **4.9.15.1 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.

#### **4.9.15.2 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.

#### **4.9.15.3 Permanente testorganisatie**

Er zijn vele vormen van permanente testorganisaties. 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 blijkt 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%**

#### 4.9.15.4 Berekeningswijze

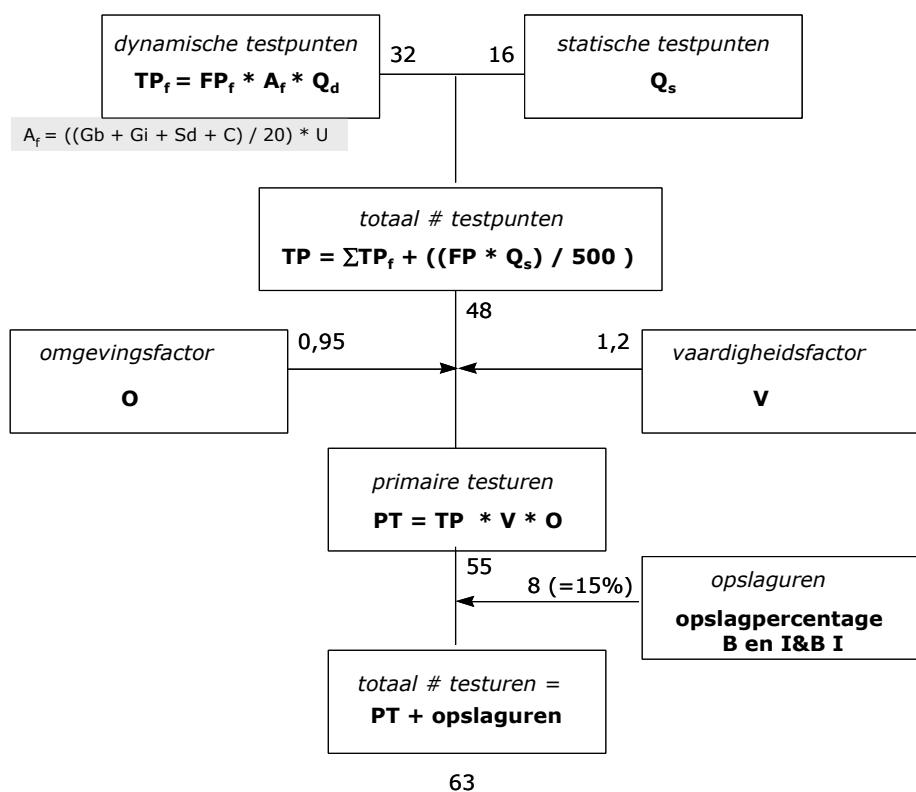
Met het opslagpercentage wordt op basis van het aantal primaire testuren de opslag (in uren) berekend.

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)
<b>Totaal aantal uren</b>	$55 + 8 = 63$

In figuur 73 is het TPA rekenvoorbeeld in zijn geheel weergegeven.



Figuur 73. Schematische weergave rekenvoorbeeld.

#### 4.9.16. 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%

#### **4.9.17. 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 functiepunktanalyse uit
- bepaal het aantal functiepunten door het aantal TOM te bepalen.

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.

### **4.10 Metrieken**

#### **4.10.1. 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. 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.

## **4.10.2. 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 [Basi 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.

### **4.10.2.1 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 evalueren, 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.

Voorbeeld

Doelstelling: Geef inzicht in de kwaliteit van het testobject

#### 4.10.2.2 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

Doelstelling: Geef inzicht in de kwaliteit van het testobject



Hoeveel fouten zijn er gevonden tijdens het testproces?

Hoeveel fouten komen er in productie voor? (tot 3 maanden na inproductie-name)

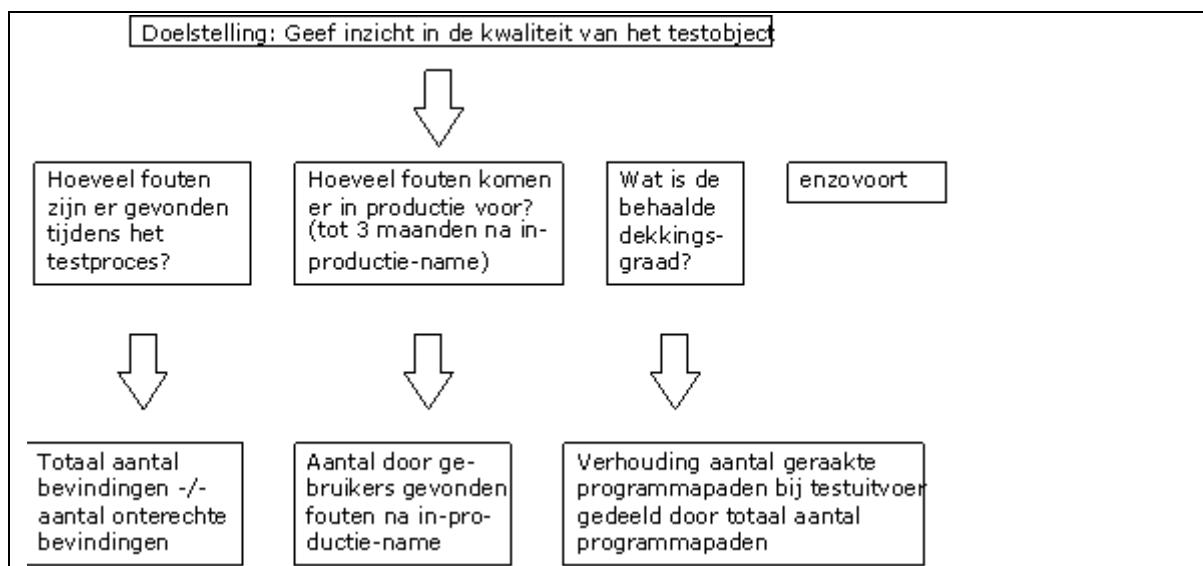
Wat is de behaalde dekkingsgraad?

enzovoort

#### 4.10.2.3 Stap 3: Van vragen naar metrics

Vanuit de gestelde vragen worden de bijbehorende metrics afgeleid. Deze metrics vormen gezamenlijk 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?

#### **4.10.2.4      *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.

#### **4.10.2.5      *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.

#### **4.10.2.6      *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.

#### **4.10.3. Hints en Tips**

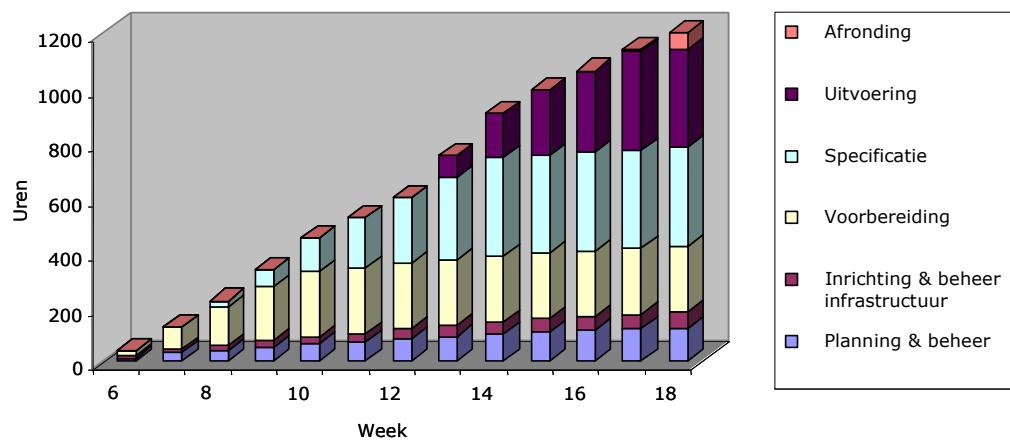
Bij het verzamelen van metrics moet de testmanager met de volgende zaken rekening houden:

- 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.

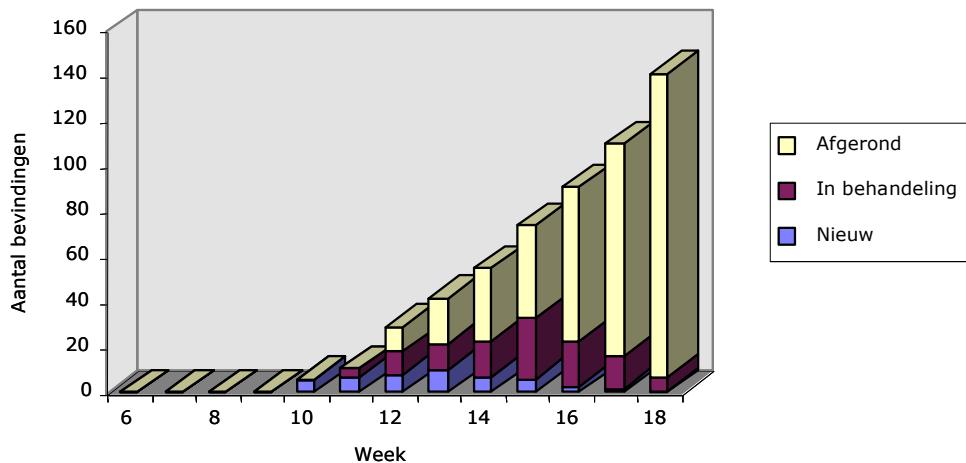
#### **4.10.4. Praktische beginset testmetrics**

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 "Rapporteren" zijn een aantal specifieker 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 75). 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 testsheets 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 paragraaf 4.7 "Bevindingen" is beschreven hoe een bevindingenadministratie ingericht kan worden. Een voorbeeld van deze rapportagevorm is hierna weergegeven (zie figuur 73):



Figuur 74. Voorbeeld bestede uren testproces per TMap-fase.



Figuur 75. 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 testscrip
- 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

$$\begin{aligned} \text{DDP ST (nadat de AT is uitgevoerd)} &: (100 / 100+60 ) = 63\% \\ \text{DDP ST (na 3 maanden productie)} &: (100 / 100+60+40 ) = 50\% \\ \text{DDP AT (na 3 maanden productie)} &: (60 / 60+40 ) = 60\% \\ \text{DDP 'overall'} &: (100+60 / 100+60+40 ) = 80\% \end{aligned}$$

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.

#### **4.10.5. 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

## **4.11 Toetstechnieken**

### **4.11.1. Inleiding**

Het resultaat van een toetsing is in belangrijke mate afhankelijk van de houding van de toetsers(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 deze paragraaf 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.

### **4.11.2. Toetsen toegelicht**

#### **4.11.2.1 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.

##### **Definitie**

Toetsen is het beoordelen van producten in het systeemontwikkelproces zonder het uitvoeren van software.

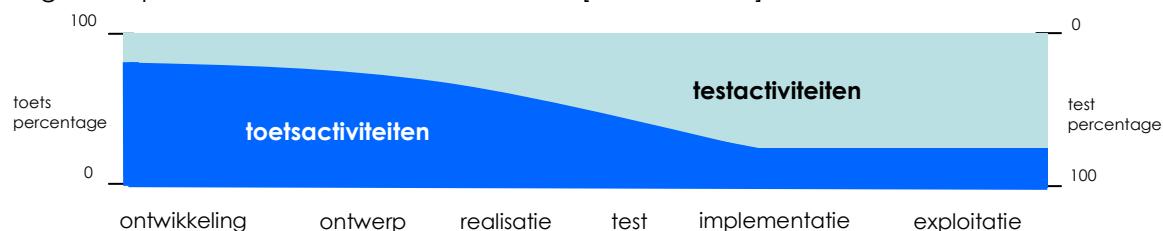
Hierbij hoeft het toetsen op tussenproducten zeker niet beperkt te blijven tot de ontwikkeldocumenten. Er kan op alle niveaus van documentatie worden getoetst:

- functioneel-/technisch ontwerp
- requirementsdocument
- managementplan
- ontwikkelplan
- testplan

- onderhoudsdocumentatie
- gebruikers-/installatie handleiding
- software
- release note
- testontwerp
- testscript
- prototype
- screen print
- enzovoort.

#### 4.11.2.2 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 76). Bij toetsen worden immers tussenproducten beoordeeld en niet, zoals bij testen, de eindproducten. En hoe eerder een fout wordt gevonden, hoe eenvoudiger en goedkoper een fout kan worden hersteld [Boehm, 1981].



Figuur 76. 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 "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 paragraaf 4.7 "Bevindingen".

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:

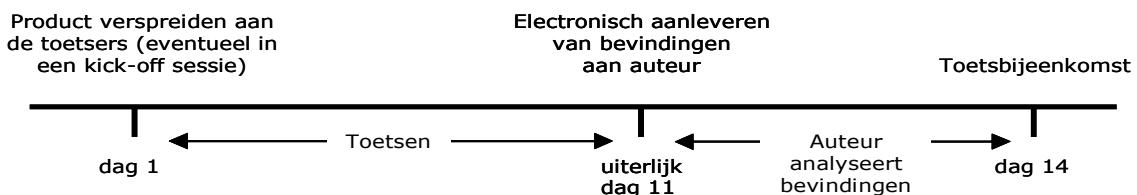
- Inleiding  
Beschrijving van het doel van de techniek en de producten waarop de techniek kan worden toegepast.
- Verantwoordelijkheden  
Beschrijving van welke rollen aan deelnemers worden toegekend.
- Entrycriteria  
Beschrijving van de noodzakelijke producten en voorwaarden waaraan voldaan moet zijn 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 77 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 77. Mogelijke planning toetstraject.

In de planning (figuur 77) 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).

### **4.11.3. Inspecties**

Er bestaan diverse invullingen van het inspectieproces. In deze paragraaf wordt een algemene invulling gegeven. Voor een specifieke, meest gebruikte, invulling wordt verwezen naar [Gilb, 1993] en [Fagan, 1986].

#### **4.11.3.1 Inleiding**

Bij het uitvoeren van een inspectie wordt een formele werkwijze gevuld, waarbij producten nauwkeurig worden gelezen door een groep deskundigen. Dit kan elk van de eerder in deze paragraaf 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.

#### **4.11.3.2 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 leesttechniek 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].

### **4.11.3.3 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.

### **4.11.3.4 Procedures**

Het inspectieproces kan worden verdeeld in de fasen planning, kick-off, voorbereiding en uitvoering:

#### **(1) 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.

## **(2) 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.

## **(3) 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.

## **(4) 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.

## **(5) 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.

#### **4.11.4.     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.

##### **4.11.4.1     Inleiding**

Bij het uitvoeren van een review wordt een formele werkwijze gevuld, 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.

##### **4.11.4.2     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.

##### **4.11.4.3     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.

#### **4.11.4.4 Procedures**

Het reviewproces kan worden verdeeld in de fasen planning, voorbereiding en uitvoering:

##### **(1) 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.

##### **(2) 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.

##### **(3) 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.

##### **(4) 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.

#### **4.11.5. 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.

### **4.11.5.1 Inleiding**

Een walkthrough kan voor alle genoemde documenten, die voor 50% tot 100% gereed zijn, worden gehouden.

### **4.11.5.2 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 geïdentificeerde acties vast. Hiernaast worden bijvoorbeeld ook bevindingen (o.a. tegenstrijdigheden, vragen en omissies) en aanbevelingen van de deelnemers genoteerd.
- 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.

### **4.11.5.3 Entrycriteria**

De walkthrough kan worden gestart als:

- Het doel van de walkthroughs duidelijk is.
- Het onderwerp (product) van de walkthrough beschikbaar is.

### **4.11.5.4 Procedures**

Het walkthrough proces kan worden verdeeld in de fasen planning, voorbereiding en uitvoering:

#### **(1) 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.

#### **(2) 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'.

### **(3) 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.

### **(4) 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.

#### **4.11.6. Keuzematrix toetsen**

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 toetsen is te geven en in welke situatie een bepaalde techniek het best is te gebruiken. Maar wellicht kan onderstaande tabel 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 presentatienvorm.
Voorbereiding	Strakke sturing op welke aspecten door de inspecteurs moeten worden beoordeeld. 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 kennismaken van het tussenproduct tot het aanleveren van bevindingen.
Productstatus en -omvang	Product is 100% gereed, nog niet definitief en beperkt van omvang (10-20 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),	Subjectief, mogelijke	Kans op ad hoc discussies,

Aspect	Inspectie	Review	Walkthrough
	relatief lange doorlooptijd.	verstoring collegiale verhoudingen.	omdat deelnemers vaak onvoorbereid zijn.

## 4.12 Kwaliteitsmaatregelen in de ontwikkelfase.

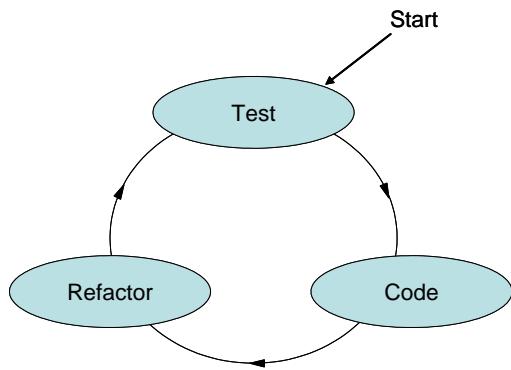
In deze paragraaf staat een aantal maatregelen beschreven die gebruikt kunnen worden in of van invloed zijn op het ontwikkelen en het ontwikkeltesten. De meeste van deze maatregelen hebben gevolgen voor de wijze waarop de unittest en/of de unitintegratietest worden uitgevoerd, behalve de codereview die 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 4.8 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.

### 4.12.1.1 Test-Driven Development (TDD)

Test-Driven Development (testgedreven software ontwikkeling), afgekort TDD, is één van de best practices van Extreme Programming (XP) (zie figuur 78). 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.



Figuur 78. Test Driven Development.

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).

#### 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 opruimen 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 unitests 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 gevuld:

- 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.

## **4.12.1.2      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. Regelmäßig 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.

### 4.12.1.3 Codereview

Een volgende maatregel om de kwaliteit van de ontwikkelde producten te verhogen is een toetsactiviteit: de codereview.

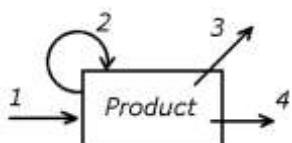
#### Definitie

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 toetsactiviteit 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 79:

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 79. De 4 soorten reviewdoelen en -vragen.

Voor de review kunnen verschillende technieken gehanteerd worden, zie ook paragraaf 4.11 "Toetsmethoden".

### 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 4.8. Deze tools worden steeds vaker opgenomen in het geautomatiseerde integratieproces, waarbij ze allerlei 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.

#### 4.12.1.4 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.

#### 4.12.1.5 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 explicet 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.

	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.
- Testcoverage tools (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.

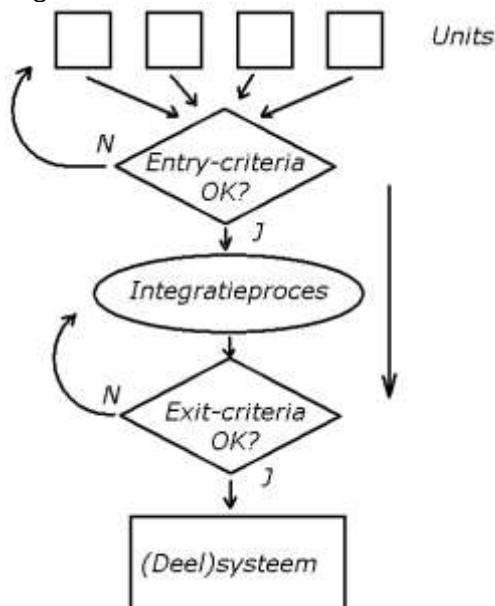
#### **4.12.1.6      Applicatie-integrator aanpak**

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 structureren 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 80). 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 80. 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.

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.

## 4.13 Beschrijven mastertestplan

### 4.13.1. Inleiding

In paragraaf “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.

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<sup>®</sup> 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.

Uitgediept

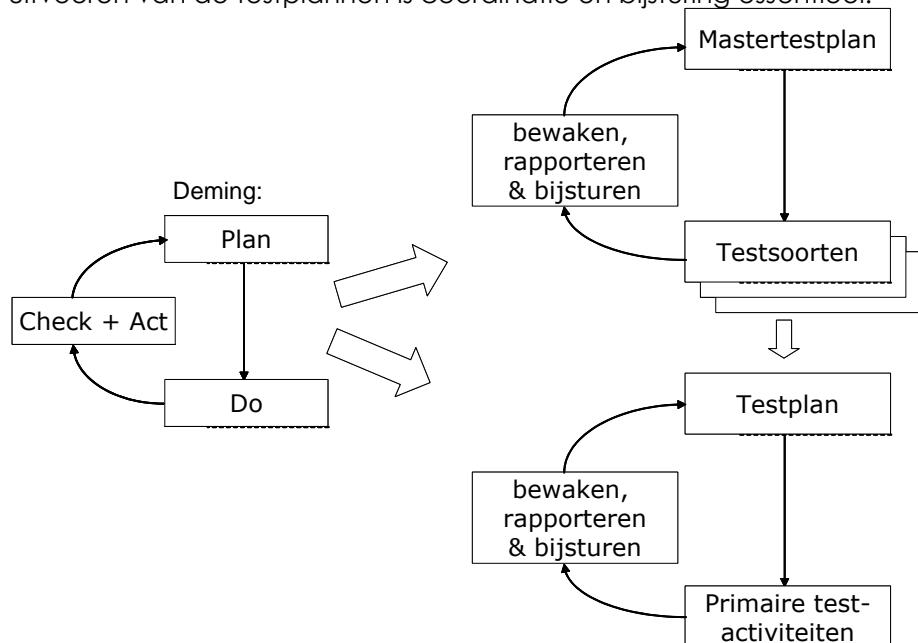
“Onnodig dubbel getest”

---

<sup>2</sup> Rational Unified Process is een geregistreerd handelsmerk van IBM

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.

Deze paragraaf 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.



Figuur 81. Deming wiel.

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 81

# Literatuur

## Aanvullende literatuur

- B Koomen, T., Aalst, L. van der, Broekman, B., Vroon, M.  
**TMap® Next, voor resultaatgericht testen**  
Vianen, Sogeti Nederland BV, 2014  
ISBN 9789075414790 (ePub 9789075414448)  
**NB Dit boek is eerder verschenen onder dezelfde naam bij**  
Uitgeverij Tutein Nolthenius, 2007
- C Boersma, A., Vooijs, E, Veltman, T.  
**Neil's Quest for Quality**  
A TMap® HD Story  
Vianen, Sogeti Nederland BV, 2014 ISBN 9789075414837
- D Aalst, L. van der; Baarda, R; Roodenrijs, E; Vink, J; Visser, B;  
**TMap NEXT Business Driven Test Management**  
s-Hertogenbosch: Uitgeverij Tutein Nolthenius, 2008  
ISBN 9789072194923
- E Aalst, L. van der; Davis, C.  
**TMap NEXT®, in SCRUM**  
Vianen, Sogeti Nederland BV, 2013  
ISBN 9789075414646  
E-ISBN 9789075414653

## Verwijzingen naar Literatuur in de tekst

- [Aalst, 1999]  
Aalst, L. van der, Koning, C. de, **Testen duur? Niet testen is duurder!**, Informatie oktober 1999, [www.informatie.nl](http://www.informatie.nl)
- [Abraan, 2003]  
Abraan, A. (2003), **Implementation Guide for ISO/IEC 19761:2003 (Measurement Manual)**, version2.2, Common Software Measurement International Consortium, [www.cosmicon.com](http://www.cosmicon.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
- [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: 9440 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
- [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
- [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
- [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
- [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 functiepuntenanalyse**, 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
- [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](http://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 [www.w3c.org](http://www.w3c.org), **Web Content Accessibility Guidelines** 1.0 en 2.0