Testen gezien vanuit een IRM-perspectief

E. Spaans CISA

Het belang dat wordt toegedicht aan testen is in de laatste jaren sterk toegenomen. De risico's bij testen worden in het algemeen onderbelicht gelaten. In het artikel wordt, met name ten behoeve van de IRM-consultant, beschreven wat testen inhoudt, waar de specifieke testrisico's aanwezig kunnen zijn en welke maatregelen hiervoor kunnen worden getroffen. Daarnaast wordt uitgebreid ingegaan op de rol die de IRM-consultant bij testen kan spelen.

Inleiding

Het belang dat wordt toegekend aan testen van toepassingsprogrammatuur lijkt de afgelopen jaren te zijn toegenomen. De millennium- en europrojecten hebben aan deze toegenomen belangstelling een grote bijdrage geleverd. Verschillende bedrijven hebben zich gestort op de toegenomen vraag naar testexpertise in de markt. Testen wordt door deze aanbieders gepositioneerd als de ultieme maatregel ter voorkoming van productieproblemen. Natuurlijk wordt daarbij de nadruk gelegd op de diensten die het betreffende bedrijf biedt en de grote voordelen die deze diensten voor de klanten van dit bedrijf betekenen. De risico's bij testen die kunnen leiden tot onjuiste uitspraken omtrent de kwaliteit van informatiesystemen worden in de regel onderbelicht gelaten.

Voor de IRM- (Information Risk Management) consultant, die in toenemende mate geconfronteerd wordt met vragen uit de markt inzake testen, geldt dat hij juist dient te handelen vanuit een risicoanalytische invalshoek. Toch vertoont gerenommeerde vakliteratuur als het handboek EDP-auditing een gapende leemte bij de bespreking van testen.

Dit artikel is geschreven als handreiking voor de IRM-consultant voor het omgaan met testen. Hiertoe gaan we achtereenvolgens in op:

* het doel van testen;
* de voorwaarden voor testen;
* het testproces;
* de controle op het verloop van testen.

Vervolgens worden de risico's en mogelijke compenserende maatregelen met betrekking tot testen besproken. Ten slotte komt de rol van de IRM-consultant in dit licht aan de orde. Hierin wordt niet alleen bekeken welke verschillende rollen zijn te onderkennen, er wordt tevens aandacht geschonken aan de fasen binnen het testen waarin de IRM-consultant kan worden ingezet.

Het doel van testen

In deze paragraaf wordt om het doel van testen te verduidelijken eerst ingegaan op het waarom van het testen. Vervolgens worden de doelstellingen zoals deze in de literatuur voorkomen geëvalueerd.

Waarom testen?

In het kader van het streven naar een zo groot mogelijke betrouwbaarheid en continuïteit van de geautomatiseerde informatievoorziening is het van groot belang de correcte werking van een systeem vast te stellen en eventuele fouten in het systeem te hebben verwijderd voordat dit systeem in productie wordt genomen (d.w.z. operationeel wordt). Het uitvoeren van tests voorafgaand aan de operationele fase is een onvervangbare maatregel. Formeel dienen de gebruikersorganisatie en de verwerkingsorganisatie het opgeleverde systeem te accepteren. Deze partijen dienen dan ook vast te stellen dat het systeem voldoet aan de door hen gestelde functionele en technische kwaliteitseisen.

Doelstellingen

In de literatuur worden veel uitspraken omtrent de doelstellingen van testen gedaan:

1. ISO: activiteiten zoals meten, onderzoeken, beproeven, keuren met kalibers van één of meer kenmerken van een product of dienst en het vergelijken van uitkomsten met gestelde eisen om te bepalen of aan deze voorwaarden is voldaan.
2. Testen is een proces van plannen, voorbereiden en meten, dat tot doel heeft de kenmerken van een informatiesysteem vast te stellen en het verschil tussen de vereiste en actuele status aan te tonen. ([Pol95])
3. Voor ontwikkelaars: testen is het gericht zoeken naar en aantonen van fouten in een systeem (voorzover dit economisch verantwoord gebeurt binnen de grenzen van de noodzakelijke foutvrijheid). ([Diem99])
4. Voor gebruikers: testen is een activiteit die dient om de onbruikbaarheid van een systeem aan te tonen (voorzover dit economisch verantwoord en voldoende effectief gebeurt). ([Diem99])
5. Testen heeft niet als doel alle fouten in een systeem op te sporen. ([Diem99])
6. Doel van de acceptatietest is het door of namens de gebruikers vaststellen dat het opgeleverde informatiesysteem voldoet aan de gespecificeerde functionele eisen en kwaliteitseisen. ([Hols97])
7. De hoofddoelstelling van het testen is het reduceren van de mate van onzekerheid over de kwaliteit van een informatiesysteem. ([Koom98])
8. The objectives of testing are to find defects and to give confidence. ([Grah99])
9. Een activiteit die is ontworpen om vast te stellen of een ruw materiaal, een onderdeel, een deelsysteem of een eindproduct voldoet aan de specificatie. ([Onni93])

Uit bovenstaande uitspraken kan worden afgeleid dat het testen van systemen twee hoofddoelstellingen heeft:
* het vergelijken van een systeem met één of meer normen teneinde vast te stellen dat het systeem voldoet;
* het vinden van fouten in een systeem.

Dit sluit ook aan op de redenen voor testen zoals opgenomen in de vorige subparagraaf.

Daarnaast blijkt dat verschillende partijen verschillende doelstellingen kunnen hebben met betrekking tot testen en dat verschillende soorten tests bestaan. In de volgende subparagrafen wordt achtereenvolgens ingegaan op de doelstellingen van testen, de invloed van de verschillende belanghebbenden en de soorten tests.

De twee hoofddoelstellingen

Het vergelijken van een systeem met één of meer normen teneinde vast te stellen dat het systeem voldoet
Zowel in theorie als in de praktijk blijkt dat volledig testen onmogelijk is. Het volgende voorbeeld kan dit verduidelijken. Als een invoerveld van 10 posities lang dient te worden getest en we alle mogelijke waarden van dit veld willen invoeren en het veld accepteert slechts hoofdletters, dan kost bij een inspanning van één menssecondе per geval deze test 26^10 menssecondenn. Dit komt neer op meer dan 4 miljoen mensjaar. Als we daarnaast ook volledig willen testen dat ongeldige waarden worden afgekeurd, dan zal iedere mogelijke waarde van de ASCII- of EBCDIC-tabel moeten worden getest. In dat geval komen we in totaal op 256^10 menssecondenn. Volledigheid is dus volstrekt onhaalbaar, zelfs in de buurt komen zal niet lukken.

Zelfs indien we beschikken over de broncode van een programma is niet met zekerheid te bepalen hoe het programma zich gedraagt onder de diverse omstandigheden. Voordat het programma kan worden uitgevoerd moet deze broncode namelijk worden vertaald naar machinetaal. Dit gebeurt door een zogenaamde compiler. Een compiler is een vertaalprogramma en het is per definitie dan onmogelijk een programma (laat staan een compiler) volledig te testen. Daarbij draait het programma onder een besturingssysteem dat ook weer bestaat uit niet-volledig geteste programma's. Wel geldt hierbij dat in veel gevallen besturingssystemen en compilers weliswaar niet volledig zijn getest maar wel in grote hoeveelheden dagelijks in productieomgevingen worden gebruikt. Hierdoor gaat men in de praktijk uit van de correcte werking van deze specifieke besturingsprogrammatuur. Veelal ten onrechte; Diemer citeert uit IBM-literatuur over het MVS-besturingssysteem het volgende: Een systeem van 5.000.000 regels assembler coding:
* Na negen jaar zijn er op het moment van rapportage nog ruim 2200 bekende fouten aanwezig.
* Van deze 2200 fouten komen er 1034 voor in slechts vier procent van de programma's.

Dit betekent volgens Diemer dat complexe programma's na negen jaar nog vijf fouten per duizend regels bevatten en dat eenvoudige programma's na negen jaar nog één fout op de vierduizend regels bevatten.

Door de onmogelijkheid volledig te testen is niet met honderd procent zekerheid vast te stellen dat een systeem volledig voldoet aan de gestelde normen. In de praktijk wordt dan ook tot slot op basis van vertrouwen besloten dat het systeem voldoet aan de gestelde eisen. Het risico van een onjuist positief oordeel omtrent het systeem is dus groot.

Het vinden van fouten in systemen
Gezien het feit dat software door mensen wordt gemaakt en mensen fouten maken, kan worden geconcludeerd dat foutloze software niet bestaat. Uit een recent onderzoek van de Metagroep blijkt zelfs dat over het algemeen vrij veel fouten voorkomen in software. Onlangs werd gemeld dat het aantal getelde fouten in software in de Verenigde Staten was afgenomen tot 1,64 per duizend regels code. Buiten de Verenigde Staten werd een gemiddelde van 3,85 fouten per duizend regels code gemeten. In Amerikaanse literatuur met betrekking tot testen wordt zelfs uitgegaan van veertig tot tachtig fouten per duizend regels code na de eerste oplevering van programmatuur. Fouten zijn dus altijd aanwezig in software. De doelstelling 'het vinden van fouten in systemen' zal dan ook bij iedere test worden gehaald. Door de onmogelijkheid van het volledig testen van systemen is het echter niet mogelijk vast te stellen dat alle fouten in een systeem zijn gevonden.

Verschillende belanghebbenden

Bij een informatiesysteem zijn verschillende belanghebbenden betrokken. Deze kunnen afkomstig zijn uit de gebruikersorganisatie, de systeemontwikkelingsorganisatie en uit de verwerkingsorganisatie. Daarnaast kunnen deze belanghebbenden afkomstig zijn van verschillende afdelingen binnen deze organisaties. Verder kunnen zij uit de verschillende lagen van de organisatie komen: het strategisch management, het tactisch management, het operationeel management en de werkvloer.

Een (gedeelte van een) informatiesysteem of één of meer aspecten van een informatiesysteem kunnen van grote betekenis zijn voor de ene belanghebbende, terwijl dit geen enkele of een heel andere betekenis heeft voor een andere belanghebbende. Zo zal het voor de ene gebruiker van groot belang zijn dat de invoerschermen goed zijn opgezet, terwijl het voor een andere gebruiker van groot belang is dat een lijst tijdig beschikbaar is. Voor een werkvoorbereider op het rekencentrum zullen de doorlooptijden van batches van groot belang zijn, terwijl het voor de netwerkbeheerder van groot belang is hoeveel data per transactie over het netwerk gaat. De belangen van de verschillende betrokkenen kunnen zelfs strijdig zijn. Dit geldt met name in gevallen waarin beperkte machinecapaciteit moet worden verdeeld.

Daarbij kunnen ook de normen per belanghebbende verschillen. Zeker bij subjectieve en lastig te specificeren aspecten als gebruiksvriendelijkheid is de mens de bepalende factor. Wat voor de een acceptabel is kan voor de ander volstrekt onacceptabel zijn.

Als gevolg van de gedeeltelijke subjectiviteit van de binnen het testen te hanteren normen is het eenduidig vaststellen van de kwaliteit van een systeem niet zonder meer mogelijk. Hiermee kan op twee manieren worden omgegaan. Ten eerste kan kwaliteit worden gedefinieerd als 'client satisfaction'. Hierbij treden problemen op in het geval dat eisen van verschillende betrokkenen strijdig zijn. Voor ieder aangetroffen item waarover de betrokkenen een verschil van mening hebben dient dan besluitvorming plaats te vinden omtrent de vraag of een fout is opgetreden of niet. Ten tweede kan kwaliteit worden gedefinieerd als 'conformance to specifications'. De normering wordt hierdoor geobjectiveerd. Niet alle eisen die aan een systeem worden gesteld, zijn echter te specificeren en/of te testen. De subjectiviteit tijdens testen is slechts gedeeltelijk uit te bannen.

Ook bij testen geldt: de mens is de maat van alle dingen.

Verschillende soorten tests

In de literatuur komt een groot aantal verschillende testsoorten voor. Tests worden onderverdeeld in testsoorten op basis van:
* *normering:* komt de normering uit de documentatie of uit de programmacode;
* *omvang:* wordt het systeem in zijn geheel getest of slechts een gedeelte.

Normering
Bij de onderverdeling van testsoorten op basis van normering worden twee soorten onderkend: de black box test en de white box test.

Bij black box tests wordt het systeem beoordeeld op basis van de verhoudingen tussen invoer en uitvoer. Uit de specificaties van het systeem worden de testgevallen en de bijbehorende uitvoervoorspellingen afgeleid. Vervolgens worden de testgevallen uitgevoerd en worden de resultaten vergeleken met de voorspellingen. Hieruit kan worden afgeleid of het systeem conform de specificaties werkt.

Een sterk punt van black box testen is dat niet geïmplementeerde maar wel gespecificeerde functionaliteit bij deze tests kan worden ontdekt. Een zwak punt van black box testen is daarentegen dat in de programmatuur opgenomen functionaliteit die niet in de specificaties staat, bij deze tests niet wordt ontdekt. Een ander zwak punt is dat de specificaties waarvan wordt uitgegaan bij een black box acceptatietest op een hoger abstractieniveau zijn beschreven dan de programmacode. Binnen de gestelde specificaties is de programmeur vrij om de beste technische oplossing te kiezen. Hierdoor geeft bij black box testen een honderd procent dekkingsgraad van de specificaties geen garantie op een honderd procent dekkingsgraad van de opgeleverde programmacode. Een groot aantal mogelijke fouten in de software kan hierdoor worden gemist. Knuth ([Knut71]) stelt dat vijftig procent van de executietijd van een programma is geconcentreerd in vier procent van de code. Wanneer de testgevallen zijn afgeleid van de specificaties en geen voeling hebben met de interne structuur van de programmatuur wordt slechts een beperkt aantal statements telkens weer getest.

Bij de white box tests wordt nadrukkelijk gebruikgemaakt van de kennis van de interne werking van het systeem. De uit te voeren testgevallen worden gespecificeerd op basis van de programmacode. De bijbehorende uitvoervoorspellingen worden hierbij afgeleid uit de specificaties. Een sterk punt van white box testen is dat de gehele software wordt meegenomen in de tests. Hierdoor kunnen buiten fouten in de software tevens onduidelijkheden en omissies in de specificaties worden ontdekt. Een zwakte van white box testen is dat het ontbreken van functionaliteit in de software niet wordt gedetecteerd.

Omvang
Bij de onderverdeling van testsoorten op basis van omvang worden door Kit ([Kit95]) twee categorieën tests onderkend:
* low level test;
* high level test.

De low level test bestaat hierbinnen uit twee verschillende soorten tests:
* unittest;
* integratietest.

De unittest is de test van de kleinste door een ontwikkelaar opgeleverde eenheid software. Deze test wordt ook wel moduletest of programmatest genoemd. Door Hetzel ([Hetz88]) wordt aangegeven dat deze test in de meeste gevallen door de ontwikkelaar zelf op een informele manier wordt uitgevoerd. Rapportage of een formele acceptatie ontbreekt. Door Pol wordt de programmatest gedefinieerd als de door een ontwikkelaar in een laboratoriumomgeving uitgevoerde test, die moet aantonen dat een programma aan de in de technische specificaties gestelde eisen voldoet.

Volgens Diemer bestaat de unittest uit een white box en een black box gedeelte. De doelstelling van de white box unittest is het aantonen van de robuustheid van het programma. De black box unittest moet kunnen worden uitgevoerd zonder hinder van technische problemen in het programma. De doelstelling van de black box unittest is volgens Diemer het aantonen dat het programma werkt conform de specificaties. De black box test wordt hierbij voor dialoogprogrammatuur onderverdeeld in een functionele test en een test van de mens-machine-interface. De test van de mens-machine-interface bestaat uit een test van de geprogrammeerde controles op de invoerschermen. De functionele test bestaat uit een test van de achterliggende functionaliteit met acceptabele invoer.

De integratietest is een test die door Pol wordt gedefinieerd als de door een ontwikkelaar in een laboratoriumomgeving uitgevoerde test, die moet aantonen dat een logische serie programma's aan de in de technische specificaties gestelde eisen voldoet.

Volgens Diemer bestaat de integratietest uit een white box test en een black box test. De white box integratietest heeft als doelstelling het aantonen van de robuustheid van het te integreren (deel)systeem. De black box integratietest moet kunnen worden uitgevoerd zonder hinder van technische problemen in het (deel)systeem. De black box integratietest heeft als doel het aantonen van de juiste doorgave van de gegevens door het (deel)systeem: de aansluiting van de gedefinieerde interfaces tussen de verschillende systeemdelen.

Volgens Hetzel is de integratietest onderdeel van de systeemtest. Meestal is deze test een black box test omdat er geen tijd is voor studie van de code. Verder is deze test formeler opgezet. De opzet van testgevallen en het onderhoud hiervan zijn beter geregeld dan binnen de unittest.

De high level test bestaat afhankelijk van de auteur die de beschrijving geeft uit verschillende soorten tests.

Volgens Kit bestaat deze test uit:
* *usability testing:* hierbij wordt het systeem getest in combinatie met de gebruiker.
Belangrijke aspecten zijn:
– efficiëntie (kan de gebruiker zijn werk doen met een minimum aan inspanning);
– toegankelijkheid (kan de gebruiker gemakkelijk navigeren binnen het systeem);
– begrijpelijkheid (begrijpt de gebruiker de structuur, de helpfunctie en de documentatie van het systeem);
* *function testing:* het doel van deze test is het ontdekken van discrepanties tussen de functionele specificaties en het gedrag van het systeem;
* *system testing:* hierbij wordt getest of het systeem voldoet aan de gestelde eisen.
Belangrijke tests hierbinnen zijn:
– volume testing (kan het programma de vereiste volumes van data en requests aan);
– load/stress testing (kan het programma piek-loadcondities aan binnen de vereiste tijdspanne);
– security testing (wordt voldaan aan de security requirements);
– performance testing (voldoet het systeem aan de eisen betreffende performance onder normale condities);
– resource usage testing (gebruikt het programma te veel resources zoals memory, disk space, etc.);
– configuration testing (functioneert het systeem naar behoren als de software en/of hardware juist is geconfigureerd);
– compatibility/conversion testing (voldoet het systeem aan de gestelde compatibiliteitseisen en werken de conversieprocedures naar behoren);
– installability testing (werken de installatieprocedures naar behoren);
– recovery testing (voldoet het programma aan de vereisten voor wat betreft de recovery);
– serviceability testing (voldoet het programma aan de eisen betreffende het beheer);
– reliability/availability (voldoet het systeem aan de gestelde eisen betreffende betrouwbaarheid en beschikbaarheid);
* *acceptance testing:* het doel van deze test is het overtuigen van de gebruiker dat het systeem gebruiksklaar is.

Binnen deze test is een onderscheid te maken tussen alfatests (door gebruikers binnen het bedrijf dat de software levert) en bètatests (door gebruikers binnen klantorganisaties).

Alle tests behalve de bètatest worden uitgevoerd door medewerkers van de leverancier van de software.

Volgens Pol bestaat high level testing uit:
* *de systeemtest:* door een ontwikkelaar in een (goed beheersbare) laboratoriumomgeving uitgevoerde test, die moet aantonen dat het systeem of delen daarvan aan de in de functionele en technische specificaties gestelde eisen voldoen;
* *de acceptatietest:* door de gebruikers en beheerders in een zo echt mogelijke productieomgeving uitgevoerde test, die moet aantonen dat het ontwikkelde systeem aan de functionele en kwalitatieve eisen voldoet. Deze acceptatietest bestaat uit twee delen: één door de beheerder en één door de gebruiker.

Volgens Diemer heeft de systeemtest als doel het vaststellen dat het gedrag van het systeem overeenkomt met de specificaties. De systeemtest wordt onderverdeeld in:
* *de functietest:* gedraagt iedere functie binnen het systeem zich volgens de specificaties;
* *de systeemintegratietest:* de juiste doorgave van gegevens van en naar overige systemen;
* *de proefproductietest:* de kwaliteit van de gebruikers- en beheerprocedures en de performance en capaciteit van het systeem.

Volgens Diemer richt de acceptatietest zich op het controleren van de volgende objecten:
* het geautomatiseerd systeem:
– correctheid van de specificaties;
– correctheid van de processen;
– correctheid van de gegevensstromen;
– correctheid van de gegevensverzamelingen;
* het systeem in zijn gebruiksomgeving:
– tijdcyclus;
– organisatie;
– doelstellingen.

De doelstelling hierbij is niet alleen het vaststellen dat het systeem overeenkomt met de specificaties maar tevens dat het systeem aansluit bij de behoeften van de gebruikers en bij de overige activiteiten van de organisatie.

In Nederlandse literatuur wordt veelal uitgegaan van organisaties met een eigen systeemontwikkeling, in de Amerikaanse van externe softwareleveranciers.

Niet in alle gevallen is duidelijk waar de verantwoordelijkheden liggen en wat op welke wijze wordt getest tijdens de verschillende soorten tests. Dit ligt gedeeltelijk aan de verschillende uitgangspunten die de auteurs hanteren. In de Nederlandse literatuur wordt veelal uitgegaan van organisaties met een eigen systeemontwikkeling. De software wordt in deze organisaties opgeleverd door de systeemontwikkelingsafdeling, getest door de

gebruikersorganisatie en de verwerkingsorganisatie en vervolgens in productie genomen. In de Amerikaanse literatuur gaat men meer uit van externe softwareleveranciers. Deze organisaties dienen hun pakketten voorafgaand aan de verkoop al uitvoerig te hebben getest, om zodoende zeker te stellen dat ze een verkoopbaar product hebben. De acceptatietest door de gebruikersorganisatie en de verwerkingsorganisatie wordt door deze organisaties veelal gezien als een informele herhaling van een gedeelte van de systeemtest.

Voor de low level tests is men het er over het algemeen over eens dat het testen van programmatuur door degene die deze heeft gebouwd niet wenselijk is. Wel wordt deze manier van testen gezien als een efficiëntieverhogende en daardoor acceptabele maatregel. Men gaat er daarbij ten onrechte van uit dat de high level tests een adequaat vangnet vormen.

De voorwaarden voor testen

In deze paragraaf worden de voor het testen belangrijkste noodzakelijke voorwaarden behandeld. Achtereenvolgens wordt ingegaan op de testomgeving, het te testen systeem (het testobject), de tester en de beschikbare hulpmiddelen.

De testomgeving

Vaak wordt bij het testen van een systeem gekozen voor een afbakening op het niveau van de applicatie. De specificaties waartegen wordt getest, beschrijven vrijwel altijd slechts de toepassing. Het te testen systeem is dan gelijk aan de te testen applicatieprogrammatuur. Wat daadwerkelijk wordt gemeten bij testen is hoe het geheel van hardware, besturingssysteem en applicatieprogrammatuur zich gedraagt. Het is echter zeer wel mogelijk dat onverwachte gedragingen van het systeem niet worden veroorzaakt door fouten in de applicatieprogrammatuur maar door onvolkomenheden in de besturingsprogrammatuur, de compiler of de onderliggende hardware (de infrastructuur). Indien we prijs stellen op een goede voorspelling van de gedragingen van het systeem in de productieomgeving dienen we bij het inrichten van een testomgeving dus zorg te dragen voor zo veel mogelijk gelijkenis met de infrastructuur in de productieomgeving.

Indien we prijs stellen op een goede voorspelling van de gedragingen van het systeem in de productieomgeving is het naast de inrichting van de testomgeving met betrekking tot de infrastructuur tevens van belang de test uit te voeren bij een zo veel mogelijk productiegelijkende belasting van het systeem als geheel. Bij belasting kan men hierbij denken aan aantallen on-linetransacties en batchjobs van zowel het te testen systeem als overige productiesystemen.

> Het is voor een leek onmogelijk de kwaliteit van de tester vast te stellen.

Een ideale inrichting van testomgevingen is in de praktijk meestal niet haalbaar. Veelal wordt op basis van een kosten-batenafweging besloten in hoeverre de testomgeving productiegelijkend wordt ingericht. Het is van groot belang de risico's in te schatten die gepaard gaan met een op bepaalde punten niet-productiegelijkend testsysteem. Testen op een niet-productiegelijkend systeem kan leiden tot een onjuist oordeel omtrent het functioneren van het systeem.

Het testobject

Door het testen van het systeem zullen onvolkomenheden worden gevonden. Omdat gewaarborgd dient te worden dat het systeem in productie naar behoren zal functioneren, zullen reparaties worden uitgevoerd op het systeem. Het gerepareerde systeem moet vervolgens weer worden getest. Door deze acties zal het testobject gedurende het testen aan wijzigingen onderhevig zijn en zullen testgevallen op verschillende tijdstippen verschillende resultaten laten zien. Bij sommige testgevallen volgens verwachting – het systeem is dan op dat punt adequaat gerepareerd –, bij andere gevallen onverwacht – bij de reparatie van een onvolkomenheid is een nieuwe geïntroduceerd. Het doorvoeren van wijzigingen op het te testen systeem in de testomgeving dient dus beheerst plaats te vinden. Vervolgens zal in beginsel de gehele test opnieuw moeten worden uitgevoerd. In de praktijk wordt om efficiëntieredenen veelal vooraf een inschatting gemaakt van de invloed van de aangebrachte wijzigingen en de noodzakelijk nogmaals uit te voeren tests.

Het testobject is gedurende het testen niet statisch. Teneinde zorg te dragen voor een betrouwbare test dient een adequaat change management met betrekking tot het testobject te worden uitgevoerd. Daarnaast geldt dat iedere inschatting van de invloed die de doorgevoerde wijzigingen hebben, risico's met zich meebrengt van een niet-gedetecteerde ongewenste verandering binnen het systeem.

De tester

Het grote belang van het testen in het kader van het waarborgen van de betrouwbaarheid en continuïteit van de geautomatiseerde gegevensverwerking en de problemen met betrekking tot de onmogelijkheid van volledig testen, de omgang met verschillende belanghebbenden en de afbakening van het testobject stellen hoge eisen aan de tester als vakman.

Het oordeel van de tester komt tot stand op basis van deskundigheid en verrichte werkzaamheden. Hierbij mag worden verwacht dat de tester de noodzakelijke werkzaamheden verricht op grond van zijn eigenbelang. De vraag of hij de vereiste deskundigheid bezit om vast te stellen welke werkzaamheden noodzakelijk zijn om tot een verantwoord oordeel te komen is echter een heel andere.

In de praktijk is het voor een leek onmogelijk de kwaliteit van een tester vast te stellen. Er bestaat geen kwaliteitskeurmerk voor testers en er bestaat geen algemeen erkende opleiding voor testers. Het is bovendien maar de vraag in hoeverre sprake is van standaardkwaliteiten die

voor iedere tester gelden. Voor het uitvoeren van een programmatest worden in de regel andere eisen gesteld dan voor een gebruikersacceptatietest, en voor een productieacceptatietest gelden weer andere eisen. Het 'vak' testen is verder relatief nieuw en er is weinig literatuur beschikbaar.

Geautomatiseerde hulpmiddelen

Leveranciers van geautomatiseerde hulpmiddelen (testtools) laten het in veel gevallen voorkomen alsof de inzet hiervan de oplossing is voor alle problemen die zich voordoen bij het testen van systemen. Door de inzet van testtools kunnen binnen dezelfde tijd met minder mankracht meer tests worden uitgevoerd. Hierdoor neemt volgens de leveranciers zowel de efficiëntie als de effectiviteit van het testen toe.

Testtools kunnen in de volgende categorieën worden onderverdeeld:
* *static analysers:* analyseren de code van programmatuur;
* *testontwerptools:* geven ondersteuning bij het ontwerpen van tests;
* *simulatoren:* simuleren de gedragingen van bijvoorbeeld randapparatuur;
* *performance testers:* meten de performance van een systeem bij een in te stellen workload;
* *debughulpmiddelen:* geven de ontwikkelaar ondersteuning bij het vinden van fouten;
* *coverage analysers:* meten de dekking van uitgevoerde tests;
* *comparators:* vergelijken bestanden, schermen, databases en signaleren verschillen;
* *capture&playbacktools:* bieden de mogelijkheid tests op te nemen en vervolgens weer af te spelen;
* *testmanagementhulpmiddelen:* bieden de mogelijkheid testgevallen en de resultaten hiervan vast te leggen en de follow-up te monitoren;
* *databasetools:* bieden de mogelijkheid bestanden en databases te manipuleren;
* *testgeneratoren:* genereren testgevallen uit broncode of specificaties.

Veelal zijn deze hulpmiddelen niet specifiek gericht op testen, maar op ondersteuning van de systeemontwikkelaars of exploitanten.

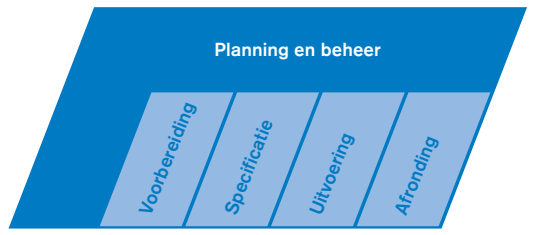
In de praktijk blijkt dat het gebruik van testtools veelal extra tijd kost. Het leren gebruiken en het installeren van testtools vergt tijd. Daarnaast geldt dat voordat met behulp van een testtool een test kan worden uitgevoerd het testgeval eerst ingebracht dient te worden. Het invoeren van een testgeval is een activiteit die minimaal evenveel tijd kost als het direct uitvoeren van deze test. Testtools leveren pas efficiëntieverhoging op als een testgeval moet worden herhaald. Ook geldt bij het gebruik van testtools dat deze minder flexibel zijn dan mensen. Kleine veranderingen in bijvoorbeeld een user-interface die door een menselijke gebruiker niet zouden zijn opgemerkt, kunnen leiden tot het constateren van fouten of het niet kunnen uitvoeren van bepaalde tests. Ten slotte geldt dat testtools programma's zijn en dus fouten kunnen bevatten. Het gebruik van testtools kan efficiëntieverhogend werken maar lost geen problemen op.

Het testproces

Wat is testen?

Door Myers werd in 1979 testen gedefinieerd als: 'het uitvoeren van een programma teneinde fouten in dat programma aan het licht te brengen'. Door verschillende andere auteurs wordt deze definitie als te eng beschouwd. Teneinde te bepalen dat een fout is opgetreden, dient namelijk eerst te worden vastgesteld wat goed en fout is. Ook deze acties maken deel uit van de activiteit testen.

In dit artikel wordt testen gezien als een onderzoek naar de kwaliteit van een geautomatiseerd informatiesysteem. Testen verloopt in beginsel dan ook volgens eenzelfde fasering als ieder ander onderzoek. Achtereenvolgens wordt een soll-positie vastgesteld, de ist-situatie wordt waargenomen, deze wordt vergeleken met de soll-positie en uit deze vergelijking wordt een conclusie getrokken en kenbaar gemaakt. Wel is binnen het testen een aantal extra voorbereidende en afrondende activiteiten nodig omdat voor het testen eerst een te testen situatie dient te worden opgezet en deze na het testen mogelijk weer dient te worden afgebroken. Het gaat hierbij vooral om het opzetten van een afgescheiden testomgeving. De bekendste methode voor testen in Nederland is TMAP. TMAP gaat uit van de volgende fasering (figuur 1):
1. planning en beheer;
2. voorbereiding;
3. specificatie;
4. uitvoering;
5. afronding.



Figuur 1. De TMAP-fasering.

In de volgende subparagrafen wordt ingegaan op de uit te voeren activiteiten per fase.

Planning en beheer
In deze fase wordt gestart met de formulering van de opdracht. Hierin worden onder andere de opdrachtnemer, de opdrachtgever, het beschouwingsgebied, het doel van het testproces, de op te leveren producten en de te testen aspecten opgenomen. Vervolgens vindt een initiële globale oriëntatie plaats waarin inzicht wordt verkregen in de projectorganisatie en de doelstellingen van het systeemontwikkelingstraject. Vooraf moet duidelijk zijn wat dient te worden opgeleverd en getest en waar eventuele risico's aanwezig zijn. In de strategiebepaling wordt vastgesteld hoe de test wordt aangepakt. Deze strategiebepaling is slechts van belang bij high level tests. Bij low level tests zijn de eenheden klein genoeg om in het geheel te worden getest (unittest) of bepaalt de integratiestrategie direct de volgorde waarin wordt getest.

Vervolgens wordt een globale begroting opgesteld. Tevens wordt geïnventariseerd welke testorganisatie en testinfrastructuur benodigd zijn en wordt een globale planning voor het testen opgesteld.

Voorbereiding
In de voorbereidingsfase wordt de testbasis (de documentatie waaruit blijkt aan welke eisen het systeem dient te voldoen, de theoretische soll-positie) gecontroleerd op testbaarheid. Aandachtspunten hierbij zijn onder andere eenduidigheid, leesbaarheid en detailniveau. Vervolgens wordt het te testen systeem onderverdeeld in testbare eenheden. Daarnaast wordt, indien noodzakelijk, de benodigde testinfrastructuur nader gespecificeerd.

Specificatie
Tijdens de specificatiefase worden per onderkende testeenheid aan de hand van de testbasis de testgevallen vervaardigd. Tijdens dit proces wordt ook de inhoud van de benodigde gegevensbestanden gespecificeerd. Vervolgens worden deze testgevallen vertaald naar concrete fysiek uit te voeren testacties. Deze testacties worden dan geclusterd in testscripts, waarna de testscripts in een testdraaiboek worden geplaatst. Daarnaast wordt tijdens deze fase de testinfrastructuur gerealiseerd.

Het specificeren van testgevallen kan volgens Pol met behulp van een aantal technieken plaatsvinden, te weten:
* algoritmetest;
* beslissingstabellentest;
* dataflowtest;
* elementaire vergelijkingen-test;
* error guessing;
* gegevenscyclustest;
* programma-interfacetest;
* procescyclustest;
* real life test;
* semantische test;
* syntactische test.

Opvallend aan deze technieken is dat het daadwerkelijk specificeren van testgevallen daarin neerkomt op twee technieken:
* het bepalen van equivalentieklassen;
* grenswaardenanalyse.

Het bepalen van equivalentieklassen is het verdelen van alle mogelijke waarden van variabelen in klassen die equivalent zijn, dat wil zeggen dezelfde gedragingen veroorzaken binnen het te testen systeem. Vervolgens worden per klasse één of meer van de mogelijke waarden gekozen als testgeval.

Grenswaardenanalyse komt neer op het per equivalentieklasse bepalen van de minimale en maximale waarde en het vervolgens testen van de grenzen. Hierbij worden het maximum binnen de klasse en de eersthogere waarde getest en het minimum in de klasse en de eerstlagere waarde.

Verder zijn geen formele testspecificatietechnieken voorhanden. Testen vindt plaats op basis van gevoel, ingegeven door de kennis en ervaring van de tester.

Extra aandacht verdient het opnemen van uitvoervoorspellingen bij testgevallen. Hoewel dit in de literatuur als absoluut noodzakelijk wordt gezien, want ‘the eye sees what it wants to see’ en ‘if there are no expectations there can be no surprises’, komt het in de praktijk vaak voor dat dergelijke voorspellingen niet worden toegevoegd.

Uitvoering
De daadwerkelijke uitvoering van de test begint met de installatie van het testobject in de testomgeving. Vervolgens wordt een zogenoemde pretest uitgevoerd teneinde globaal te kunnen vaststellen dat het testobject van voldoende kwaliteit is om te worden getest. Indien dit het geval is worden de gegevensbestanden in de testomgeving gevuld met de initiële waarden en worden vervolgens de tests uitgevoerd.

Na uitvoering van de tests worden de resultaten van de test vergeleken met de voorspellingen en wordt onderzocht wat de oorzaak is van eventuele onvolkomenheden. Dit kan zijn een fout in de programmatuur, maar ook een fout in de infrastructuur of in het testgeval. Na herstel van de gevonden onvolkomenheden worden minimaal de betreffende tests opnieuw uitgevoerd en beoordeeld totdat alle tests succesvol zijn uitgevoerd.

Afronding
De fase afronding kent een aantal evaluaties en een conserverende activiteit. De evaluaties betreffen de kwaliteit van het testobject en de kwaliteit van het testproces. Beide evaluaties dienen als invoer voor het op te leveren evaluatierapport. Dit rapport bevat als belangrijkste onderwerp het kwaliteitsadvies met betrekking tot het testobject.

De conserverende activiteiten hebben betrekking op de zogenoemde testware. Deze testware bestaat uit testgevallen en -resultaten en beschrijvingen van de gebruikte infrastructuur. Het doel van deze activiteit is het veiligstellen van deze items ten behoeve van het testen van wijzigingen in de onderhoudsfase.

De controle op het verloop van testen

Doordat het onmogelijk is een systeem volledig te testen is het meten van het verloop van testen geen eenvoudige zaak. Hoeveel testgevallen zijn nodig om een systeem te testen of hoeveel mensuren zijn hiervoor nodig? In de praktijk wordt op basis van de volgende gegevens gemeten:
* resultaten uit het verleden (metrics);
* resultaten uit het testproces zelf;
* dekkingsgraden.

Resultaten uit het verleden

Koomen stelt dat metrics van groot belang zijn voor de beheersing van het testproces. Hierbij worden drie niveaus onderscheiden:
* per project;
* per systeem;
* per organisatie.

Op ieder niveau worden de volgende zaken gemeten:
* input;
* output;
* resultaat.

Op projectniveau worden de volgende zaken gemeten:
* input:
– gebruikte resources (in mensuren);
– uitgevoerde activiteiten (in bestede uren en doorlooptijd);
– omvang en complexiteit (in functiepunten, lines of code (LOC), bouwinspanningsuren);
* output:
– testproducten (specificaties en testgevallen, logverslagen);
– testvoortgang (uitgevoerde tests, status);
– aantal bevindingen (per testsoort, deelsysteem, oorzaak, prioriteit, status);
* resultaat:
– foutvindeffectiviteit (gevonden fouten ten opzichte van aanwezige fouten en analyse van testeffectiviteit);
– foutvindefficiëntie (het aantal fouten per besteed uur);
– testdekkingsgraad (de testdoelen die afgedekt zijn met een testgeval ten opzichte van het totaal aantal testdoelen);
– bevindingen op de testware (het aantal fouten waarvan de oorzaak verkeerd testen bleek te zijn ten opzichte van het totaal aantal fouten);
– perceptie van kwaliteit (via reviews en interviews met gebruikers, testers en andere betrokkenen).

Op basis van de gegevens uit voorgaande trajecten kan dan worden afgeleid hoeveel tijd en inspanning naar verwachting gemoeid zijn met een uit te voeren test.

Het is de bedoeling dat de resultaten van de metingen per project worden geaggregeerd tot op systeemniveau. Wel dient hierbij rekening te worden gehouden met het uiteindelijke gedrag van het systeem in productie. Als de kwaliteit van het opgeleverde systeem onvoldoende blijkt te zijn dan dient dit te worden meegenomen in de metrics (foutvindeffectiviteit). Testen in onderhoudssituaties kan hierbij ook worden meegenomen.

Op organisatieniveau kunnen vervolgens de verschillende projecten en systemen met elkaar worden vergeleken. De verschillen tussen de systemen en projecten kunnen worden geanalyseerd en de resultaten van deze analyse kunnen worden gebruikt voor procesverbetering.

Voor het adequaat kunnen meten is het zaak eenduidige definities van de te meten variabelen vast te stellen. Verder dienen eenduidige meetrichtlijnen te worden uitgevaardigd, zodat de metingen in alle gevallen hetzelfde worden uitgevoerd. Daarnaast moeten tijdens het testen opgetreden incidenten die van invloed zijn op de meetresultaten zo spoedig mogelijk worden vastgelegd zodat later bij de analyses geen verkeerde conclusies worden getrokken.

Het grootste risico bij het gebruiken van metrics is dat niet alle variabelen die van invloed zijn, worden meegenomen. Een aandachtspunt dat bij deze metrics buiten beschouwing wordt gelaten, is de systeemontwikkelingsorganisatie. Bij een van project tot project wisselende kwaliteit en/of productiviteit van de systeemontwikkelingsorganisatie is het vergelijken van de testeffectiviteit en -efficiëntie tijdens deze projecten vrijwel onmogelijk.

Resultaten uit het testproces zelf

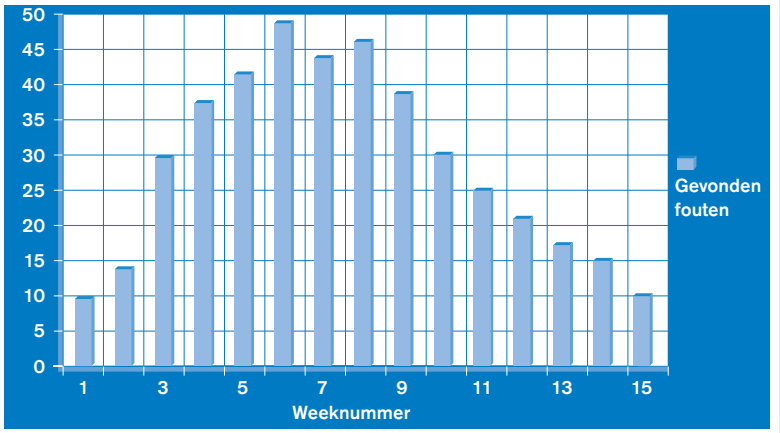
Diemer beschrijft de foutvindcurve als hulpmiddel bij de vaststelling of het testen kan worden gestopt. Een voorbeeld van een foutvindcurve is in figuur 2 opgenomen.

In de foutvindcurve wordt het aantal gevonden fouten van het te testen systeem aangetekend. Op basis van het aantal gevonden fouten per tijdseenheid kan worden besloten dat het testen in een bepaalde fase nog slechts weinig toevoegt en dus beter kan worden gestopt. Een nadeel van deze manier van meten is dat de gereserveerde testtijd verstreken kan zijn voordat het aantal fouten per tijdseenheid een acceptabel niveau heeft bereikt. Een ander groot nadeel van deze manier van meten is dat de verrichte testinspanning buiten beschouwing blijft. Binnen een testtraject zal het systeem, na enige aanlooppro-blemen, zeer veel fouten laten zien. Deze fouten dienen vervolgens te worden opgelost. Het is in veel gevallen niet zinnig om het nog niet aangepaste gedeelte van het systeem door te testen. Hierdoor kan het voorkomen dat testers een gedeelte van de tijd niet effectief kunnen zijn. Als gevolg hiervan daalt het aantal gevonden fouten in die tijd, terwijl het zeer wel mogelijk is dat achter de gevonden en gecorrigeerde fouten zich nieuwe of nog onontdekte fouten bevinden. In de tijd vlak na de oplevering van het gecorrigeerde deelsysteem kan zich opnieuw een explosie van fouten voordoen. Het is dus zeer wel mogelijk dat deze curve in de praktijk geen curve is maar een golvende lijn zoals weergegeven in figuur 3. Bovendien is niet goed in te schatten wat er in de volgende release van gecorrigeerde programmatuur aan fouten zit.

Dekkingsgraden

In de literatuur wordt een tweetal typen dekkingsgraad beschreven:
* *functionele dekkingsgraad (functional coverage):* de graad waarin de specificatie van een te testen object is beproefd door een set van testgevallen;
* *fysieke dekkingsgraad (structural coverage):* de graad waarin de uit te voeren statements van een te testen object zijn beproefd door een set van testgevallen.

Figuur 2. Een foutvindcurve.



Fysieke dekkingsgraad wordt in de literatuur onderverdeeld in:

	Statement coverage	Decision coverage	Condition coverage	Decision/ condition coverage	Multiple condition coverage
Elke programmaregel is ten minste eenmaal uitgevoerd	Ja	Ja	Ja	Ja	Ja
Elke beslissing neemt ten minste eenmaal alle mogelijke waarden aan	Nee	Ja	Nee	Ja	Impliciet
Elke conditie in een beslissing neemt ten minste eenmaal alle mogelijke waarden aan	Nee	Nee	Ja	Ja	Impliciet
Alle mogelijke combinaties van conditiewaarden in elke beslissing komen ten minste eenmaal voor	Nee	Nee	Nee	Nee	Ja

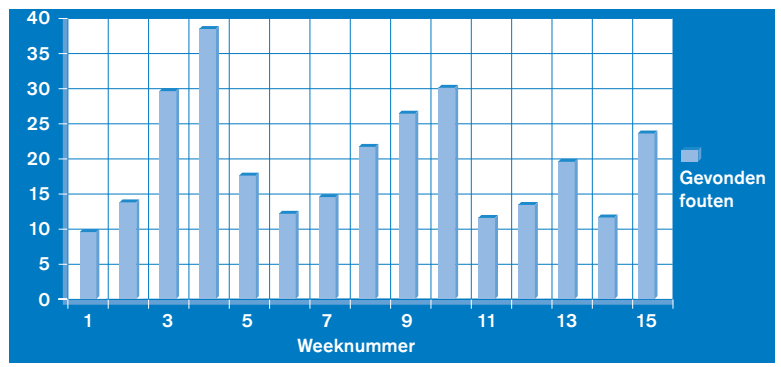
Tabel 1. Fysieke dekkingsgraden.

Als een van tevoren afgesproken percentage dekking wordt bereikt dan wordt het testproces gestopt. Hierbij kan één van beide typen dekkingsgraad worden gebruikt of een combinatie van de twee.

Een groot nadeel van het hanteren van de dekkingsgraad als stopcriterium is dat geen onderscheid wordt gemaakt tussen belangrijke en minder belangrijke functionaliteit. Wel kan een aparte dekkingsgraad per deelsysteem of zelfs per programma worden gehanteerd.

De hoogste dekkingsgraad geeft de uitvoering van de structured test. Deze methode is gebaseerd op de zogenoemde cyclomatic complexity. Cyclomatic complexity meet de hoeveelheid beslissingslogica in een module. Zij is gebaseerd op de structuur van de control flow van de module. De cyclomatic complexity van een module is precies gelijk aan het aantal mogelijke paden door dit programma. De cyclomatic complexity geeft daarmee tevens het aantal minimaal benodigde testgevallen voor deze module aan. De testgevallen zijn af te leiden uit de condities binnen de programmacode. De uitvoervoorspellingen voor de betreffende testcases dienen uit de door de gebruikersorganisatie goedgekeurde specificaties te worden gehaald. Op deze manier is gewaarborgd dat de resultaten van het programma correct worden geïnterpreteerd. Impliciet wordt hierbij een controle uitgevoerd op de juistheid en het detailleringsniveau van de specificaties en de door de programmeur uitgevoerde analyse.

Figuur 3. Een foutvindcurve is niet altijd een curve.



Dekkingsgraad wordt in de praktijk weinig gebruikt als meetmiddel. Het grote nadeel van het gebruik van dekkingsgraad als stopcriterium is dat veel testgevallen nodig zijn om een goede dekkingsgraad (honderd procent) te bereiken. Een dekkingsgraad zegt slechts iets over de hoeveelheid statements die zijn getest, niet over het belang van deze statements.

Evaluatie

In deze paragraaf wordt achtereenvolgens ingegaan op de risico's die binnen testen worden gelopen en op de mogelijke compenserende maatregelen hiervoor.

Risico's

Als gevolg van de onmogelijkheid volledig te testen bestaat te allen tijde het risico dat het geteste systeem zich in productie niet gedraagt zoals verwacht. Hierdoor kunnen de betrouwbaarheid en de continuïteit van de geautomatiseerde gegevensverwerking in gevaar worden gebracht. Het is van groot belang dat dit risico adequaat wordt beheerst en niet onnodig wordt vergroot. Dit kan worden bereikt door naar behoren te testen. De volgende factoren bemoeilijken dit:

* onvoldoende testen;
* onjuist meten;
* onjuist specificeren/interpreteren.

Onvoldoende testen
Het is onmogelijk absoluut te bepalen wanneer voldoende is getest. Volledig testen is onmogelijk en aan de beschreven methoden van meting zijn grote risico's verbonden als de van toepassing zijnde randvoorwaarden niet naar behoren zijn ingevuld. Daarbij komt dat testtrajecten vrijwel altijd onder grote tijdsdruk worden uitgevoerd. Het gevaar bestaat dat besloten wordt het testen dan maar wat in te korten. Hierdoor wordt de kans op een onterecht positief oordeel omtrent de kwaliteit van het systeem groter: hoe minder wordt getest, hoe minder fouten worden gevonden. Tevens bestaan onduidelijkheden met betrekking tot de verschillende testsoorten en wie verantwoordelijk is voor welke tests, waardoor het risico bestaat dat belangrijke tests niet worden uitgevoerd.

Onjuist meten
Doordat testen voorafgaand aan inproductiename wordt uitgevoerd in een separate testomgeving die qua inrichting en belasting veelal niet exact hetzelfde is als de productieomgeving, wordt het risico gelopen dat het systeem zich in productie niet hetzelfde gedraagt als in de testomgeving.

Daarnaast is het, doordat het te meten systeem (de applicatieprogrammatuur) niet gelijk is aan het gemeten systeem (de applicatieprogrammatuur in combinatie met het onderliggende besturingssysteem en de bijbehorende hardware), mogelijk dat onvolkomenheden onterecht worden toegewezen aan het te meten systeem.

Verder geldt dat één van de doelstellingen van testen het vinden van fouten is. Deze fouten dienen bij voorkeur te zijn hersteld voordat het systeem in productie wordt genomen. Teneinde te constateren dat dit daadwerkelijk het geval is dienen de verbeterde versies van de programmatuur ook weer te worden getest. Hierdoor verandert het testobject gedurende het testtraject. Testresultaten bereikt in het verleden zijn dan in het geheel geen garantie meer voor de toekomst. Het risico bestaat dat met de verbeterde versies van programmatuur geïntroduceerde fouten niet meer worden geconstateerd.

Onjuist specificeren/interpreteren
Doordat de verschillende belanghebbenden bij het systeem vanuit verschillende perspectieven kijken naar het systeem en daarbij een verschillend belang hechten aan (gedeelten van) het systeem, worden eisen, wensen en specificaties met betrekking tot het systeem op verschillende wijzen geïnterpreteerd en geuit. De eisen en wensen zijn niet allemaal expliciet specificeerbaar en mogelijk strijdig. Er blijft dus een subjectief element met betrekking tot de normstelling en de interpretatie hiervan tijdens de test aanwezig. Hierdoor is het mogelijk dat het geteste systeem in productie niet aan ieders verwachtingen voldoet of dat vertragingen ontstaan door wijzigingen in de programmatuur als gevolg van meningsverschillen tussen belanghebbenden.

Daarnaast geldt dat geen formele specificatietechnieken voorhanden zijn. Testen vindt vooral plaats op gevoel. Doordat dit gevoel wordt ingegeven door persoonlijke kennis en ervaring is de kwaliteit van de uitgevoerde test zelfs bij een objectieve normstelling sterk afhankelijk van de tester.

Compenserende maatregelen

Ter beheersing van de risico's bij testen is kwaliteitsborging noodzakelijk. Hiervoor zijn onder andere de volgende maatregelen voorhanden:
* een gestructureerd testproces;
* een deskundig tester.

Een gestructureerd testproces
Door het werken aan de hand van een gestructureerde methode wordt gewaarborgd dat aandacht wordt besteed aan de onderkende risico's met betrekking tot testen. In de beschrijving van de uit te voeren activiteiten dient te zijn opgenomen welke risico's kunnen ontstaan en hoe deze kunnen worden geminimaliseerd. Gestructureerde methoden zijn voldoende aanwezig. Hierbij geldt wel dat de methode geen zekerheid kan bieden. Niet alle methoden geven bovendien voldoende handvatten voor alle risico's.

Een deskundig tester
Een deskundig tester is in staat met behulp van een methode adequaat om te gaan met de binnen het testen optredende risico's. Een deskundig tester zou daarnaast door zijn kennis en ervaring in staat moeten zijn de lacunes in de verschillende methoden te herkennen en zelf maatregelen hiervoor te treffen. Een deskundig tester is echter niet direct als zodanig herkenbaar omdat een kwaliteitskeurmerk voor testers momenteel niet voorhanden is.

De rol van de IRM-consultant

Gezien de risico's die aanwezig zijn bij testen lijkt het geen overbodige luxe om een IRM-consultant in te schakelen bij testen. Hierbij kan deze verschillende rollen vervullen:
* beoordelaar;
* adviseur;
* coördinator.

De context

De door de IRM-consultant uit te voeren werkzaamheden met betrekking tot testen vinden plaats binnen de context van de risico's voor de systeemontwikkeling (het totstandkomingsproces) en het bedrijfsproces.

Binnen de systeemontwikkeling treden zoveel risico's op dat zij een apart artikel rechtvaardigen. De gevolgen zijn tweeledig:
* te late oplevering van producten;
* oplevering van producten van onvoldoende kwaliteit.

Beide factoren hebben een invloed op de uit te voeren test. Producten van onvoldoende kwaliteit leiden tot het vinden van veel fouten en daardoor veel extra tijdsbesteding als gevolg van veel uit te voeren hertests en het stilzitten van testers tijdens reparatiewerkzaamheden. Te late oplevering legt over het algemeen ook een extra tijdsdruk op het testen. Vaak is inproductiename lastig uit te stellen en moet de test snel worden uitgevoerd.

Het bedrijfsproces dat zal worden ondersteund door het te testen systeem is ook van invloed op het uit te voeren testproces. Hierbij is met name van belang in hoeverre dit bestand is tegen fouten binnen het te testen systeem, bijvoorbeeld door handmatige controles of vergelijkingen met uitvoer van andere systemen.

Binnen de geschetste context richten de werkzaamheden van de IRM-consultant zich met name op de volgende zaken:
* het testproces;
* de testproducten;
* de testorganisatie;
* de testomgeving.

Het testproces

Het uitgangspunt voor de beheersing van het testproces is het testplan. In dit testplan dient te zijn opgenomen hoe het testproces dient te verlopen. Aandachtspunten hierbinnen zijn voortgangs- en kwaliteitsbewaking.

Binnen het aandachtsgebied voortgangsbewaking wordt met name aandacht geschonken aan de planning en de begroting van het testen en de daarbij van toepassing zijnde uitgangspunten en randvoorwaarden in combinatie met de rapportages omtrent voortgang en besteed budget.

Binnen de kwaliteitsbewaking wordt aandacht geschonken aan de procedures met betrekking tot configuratie- en change management en de rapportage van incidenten en problemen.

Zonder een adequate administratie van het testproces is geen beheersing mogelijk.

Het is van groot belang dat de binnen het testplan gehanteerde procedures worden uitgevoerd en dat daadwerkelijk adequaat hieromtrent wordt geadministreerd en dat dossiers voldoende toegankelijk zijn. Zonder een adequate administratie is geen beheersing mogelijk.

De testproducten

Gedurende het testen wordt een groot aantal inhoudelijke producten vervaardigd. Ieder product bezit een aantal specifieke aandachtspunten betreffende de kwaliteit. In deze subparagraaf wordt achtereenvolgens ingegaan op:

* de risicoanalyse;
* de teststrategie;
* de testgevallen;
* de bevindingen;
* het logboek;
* het kwaliteitsadvies.

In veel gevallen is één van de eerste producten die worden opgeleverd de risicoanalyse. De risicoanalyse is de basis voor de te bepalen teststrategie en hierdoor van cruciaal belang voor de gehele test. De juistheid en de volledigheid van de risicoanalyse dienen absoluut naar behoren te zijn gewaarborgd.

De teststrategie bepaalt op basis van de uitgevoerde risicoanalyse de verdeling van de testinspanning over het systeem en de te testen aspecten. De teststrategie dient dus voldoende te zijn onderbouwd door de risicoanalyse.

De vervaardiging van testgevallen vindt grotendeels intuïtief plaats. Van groot belang is dat in ieder geval equivalentieklassen zijn bepaald en grenswaardenanalyse is toegepast. Verder is van belang dat het testgeval uniek identificeerbaar is en dat is vastgelegd wie het testgeval heeft aangemaakt, wanneer dat is gebeurd, op basis van welke documentatie en met welk doel. Daarnaast dient een uitvoervoorspelling aanwezig te zijn. In het kader van herbruikbaarheid is het van belang dat duidelijk wordt gemaakt welke systeemfunctie, scherm of lijst wordt getest.

Gedurende het testen worden bevindingen gedaan. Deze bevindingen kunnen betrekking hebben op de uitgangsdocumentatie (de normen) of op het systeem. Bij bevindingen is het van belang op te nemen wie de bevinding heeft gedaan, een korte omschrijving van de bevinding, wanneer de bevinding is gedaan en bij welk testgeval of onderzoek dit heeft plaatsgevonden.

Alle gebeurtenissen met betrekking tot het testen dienen inclusief tijdstip en betrokkene te worden opgenomen in een zogenaamd logboek. Hierbij gaat het om uitgevoerde testgevallen, incidenten, veranderingen in de testomgeving, etc. De volledigheid van dit logboek is van groot belang in het kader van probleemoplossing en evaluatie.

Het kwaliteitsadvies is het advies aan de opdrachtgever met betrekking tot de implementatiebeslissing. Dit advies dient te zijn onderbouwd door de bevindingen en het bijgehouden logboek. Daarnaast dient het een lijst te bevatten van nog niet opgeloste problemen en de workarounds. Het advies dient duidelijk de kwaliteit van de uitgevoerde test aan te geven door het aantal geplande en het aantal succesvol en niet-succesvol uitgevoerde tests te vermelden. Bij voorkeur dient een uitspraak te worden gedaan omtrent de behaalde dekkingsgraad.

De testorganisatie

Binnen de testorganisatie is het vooral belangrijk dat taken, verantwoordelijkheden en bevoegdheden adequaat zijn verdeeld. Ook is het van belang dat de juiste belanghebbenden deel uitmaken van de testorganisatie en dat de omvang van de organisatie kwalitatief en kwantitatief in overeenstemming is met de zwaarte van de test zoals die blijkt uit de risicoanalyse en de strategie.

De testomgeving

Het volledig productiegelijkend inrichten van testomgevingen is niet haalbaar. Wel dient zo veel mogelijk te worden gestreefd naar een zodanige inrichting dat de binnen de uit te voeren test van belang zijnde kwaliteitsaspecten naar behoren kunnen worden getest in de betreffende omgeving. Dit betekent dat beslissingen met betrekking tot de inrichting van de testomgeving dienen te zijn onderbouwd door de teststrategie. Verder betekent dit dat indien de testomgeving niet volledig voldoet aan de vereisten voor de uit te voeren test, de risico's hiervan terug dienen te komen in de risicoanalyse.

Literatuur

[Diem99]
E.A.P. Diemer, *Testen van informatiesystemen*, Academic Service, Schoonhoven 1999.

[Grah99]
D.R. Graham, *Measuring the value of testing*, presentatie Algemene Ledenvergadering van de vereniging TestNet, 1999.

[Hetz88]
W.C. Hetzel, *The complete guide to software testing*, QED Information Sciences, 1988.

[Hols97]
J.W. van Holst, *Het acceptatieproces beheerst*, EDP Auditor, nummer 1, 1997.

[Kit95]
E. Kit en S. Finzi, *Software testing in the real world: Improving the process*, Addison Wesley, 1995.

[Knut71]
D. Knuth, *An empirical study of Fortran programs*, Software Practise and Experience, april-june 1971.

[Koom98]
T. Koomen, M. Pol, H.W. Broeders en H. Voorthuyzen, *Test Process Improvement: A practical step-by step guide to structured testing*, Addison Wesley, 1998.

[Meye79]
G.J. Meyers, *The art of software testing*, Wiley-Interscience, 1979.

[Onni93]
A. Onnias, *Verklarende woordenlijst kwaliteitszorg*, Kluwer Technische Boeken, Deventer 1993.

[Pol95]
M. Pol, E. van Veenendaal en R. Teunissen, *Testen volgens TMAP*, Uitgeverij Tutein Nolthenius, 's-Hertogenbosch 1995.

E. Spaans CISA
is als manager werkzaam binnen de unit Management van InformatieTechnologie van KPMG Information Risk Management. Hij heeft een uitgebreide ervaring op het gebied van systeemontwikkelingsvraagstukken, kwaliteitszorg en testen.