September 2006

Nieuwsbrief van de vereniging TestNet

TNN

TestNet Nieuws

Van de redactie

Door Meile Posthuma
tnn@testnet.org

De vakantie weer achter de rug en vol energie tegen het testwerk aan. Om iedereen lekker in te laten komen is het de redactie toch gelukt om weer een lezenswaardige nieuwsbrief in elkaar te zetten. Er komt aan het eind van het jaar een boek uit op testgebied en voor een heleboel testers een belangrijk boek n.l. TMap®Next. In deze TNN kunt u alvast lezen wat er zoal nieuw is in deze uitgave. Verder hebben we artikelen over Service Oriented Architecture (SOA), een onderwerp wat steeds belangrijker wordt voor ons testers. Natuurlijk zijn er ook weer de “5 vragen aan “. Al met al weer voldoende leesvoer voor de tester.

In dit nummer

Van de voorzitter

Door Bob van de Burgt
voorzitter@testnet.org

Najaarevenement groot succes!!

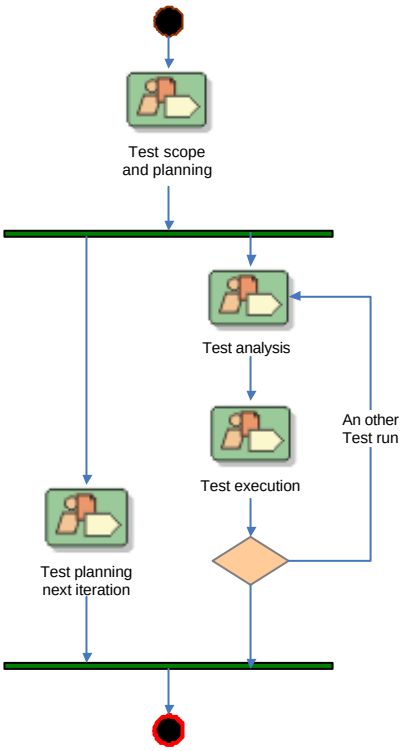
Het najaarsevenement met als thema “De Ideale Teststrategie” was een groot succes. Er waren meer dan 300 deelnemers en 15 sponsoren op af gekomen. Er stonden maar liefst twee internationale key-notes op het programma. Julie Gardiner nam ons mee in haar ervaringen met risk based testen en Thorkil Sonne gaf een presentatie over speciale skills in testen. Hij heeft een specialistisch bedrijf opgezet waarbij de capaciteiten van autistische testers optimaal worden ingezet. Een zeer indrukwekkend verhaal. Wat opviel op het evenement was de uitstekende sfeer en de prima mogelijkheden tot netwerken. Er was sprake van een soort reünie voor testers. Al met al een uitstekend evenement. Petje af voor de evenementencommissie.

De lat ligt nu uiteraard hoog voor het 10 jarig bestaan van TestNet in 2007, een lustrumjaar!! Ik kijk er nu al naar uit.

De meerwaarde van gestructureerd testen binnen RUP

Door Han Duisterwinkel,
han.duisterwinkel@logicacmg.com.
Nico van der Elst,
nico.van.der.elst@logicacmg.com.

Veel organisaties zijn voor de uitvoering van ICT projecten op zoek naar de juiste methodische aanpak. Dit om het proces beheersbaar en voorspelbaar te maken en uiteindelijk een kwalitatief goed eindproduct op te leveren. IBM’s Rational Unified Process (RUP), een methode voor softwareontwikkeling gebaseerd op best practices (key principles), sluit aan op de wens om het proces



beheersbaar en voorspelbaar te maken. Het opleveren van een kwalitatief goed eindproduct wordt binnen RUP gemeten door middel van testen. Het maximale rendement van het testen wordt opgeleverd door dit op een gestructureerde en herhaalbare manier te doen. Dit artikel gaat in op de meerwaarde van gestructureerd testen binnen RUP

Proces testdiscipline

Het testproces dat onderdeel is van de testdiscipline in RUP heeft ten opzichte van gangbare testmethodieken een aantal nadelen.
De belangrijkste zijn:

- Testanalyse mist in het testproces.
- Niet alle testsoorten zijn in het testproces geïntegreerd.
- Erg veel niveaus in testproces (te veel detail).

Naar aanleiding van bovenstaande nadelen is door de auteurs van dit artikel het testproces aangepast, zie figuur.
Het belangrijkste verschil op het hoogste niveau van het testproces is de toevoeging van de testanalyse.

Testanalyse

Het testanalyseproces is een wezenlijk onderdeel van het gehele testproces. Een kenmerk van het proces is dat de testanalyse al start als de requirements worden opgesteld. Deze opgestelde (goedgekeurde) requirements zijn de basis voor test, maar ook voor implementatie en acceptatie. Aan de hand van deze testbasis worden testscenario's (1) en testclusters (2) opgesteld die als basis dienen voor de testuitvoering.

Een belangrijk onderdeel van het testanalyseproces is het opstellen van testscenario's. Deze scenario's dienen ook als basis voor de acceptatietest. Er zijn in het testproces een gering aantal activiteiten gedefinieerd die elk een specifiek gebied beslaan. Hierdoor blijft het proces overzichtelijk. Naast de activiteiten is er voor elk artefact een template beschikbaar.

Afhankelijk van het soort project, het type systeem en bijbehorende ontwikkeltechniek kan besloten worden om delen van de test te automatiseren. Ook dit deel is geïntegreerd in het testproces. De testanalyse die voor het handmatig testen wordt uitgevoerd kan ook gebruikt worden voor het automatisch uitvoeren van de test.

Testuitvoering

De testuitvoering vindt op een gestructureerde manier plaats waarbij gebruik wordt gemaakt van de testvoorbereiding die tijdens de testanalyse is gedaan. De testscenario's en de testclusters worden hierbij uitgevoerd. Bij het testen worden de resultaten (en eventueel bijbehorende bevindingen) gestructureerd geregistreerd. Hierdoor worden niet alleen de testresultaten maar ook de kwaliteitsontwikkeling van het systeem zichtbaar.

Gestructureerd testen binnen RUP volgens LogicaCMG

Binnen LogicaCMG is een gedegen testmanagement-aanpak ontwikkeld die gebaseerd is op Risk & Requirement Based Testing (RRBT). RRBT richt zich op het identificeren van risico's welke verbonden zijn met het ontwikkelen en implementeren van een systeem. Naast de Testmanagementaanpak heeft LogicaCMG TestFrame ontwikkeld om het testproces op de juiste manier (verder) in te richten en uit te voeren.

Fasering RUP, RRBT en TestFrame

Tijdens de eerste twee fases van het project, in RUP termen: de inception- en elaboration-fase, is het zaak dat de testverantwoordelijke het testplan en de teststrategie voor het volledige testtraject in kaart brengt. De inspanning rond RRBT ligt dan ook voornamelijk in deze twee fases.

Tijdens elke fase en elke iteratie van het project zal er een deel testanalyse en testuitvoering worden gedaan. In de inception- en elaboration-fases zal de nadruk liggen op het ondersteunen en opzetten van de teststrategie en -inrichting. In de construction-fase ligt de nadruk op zowel de testanalyse als de testuitvoering. In de transition-fase ligt de nadruk op de testuitvoering. Kortom TestFrame wordt gedurende het gehele project toegepast.

Past RRBT bij RUP?

Binnen iteratief ontwikkelen volgens RUP worden in korte tijdscycli de grootste risico´s en de daaraan gerelateerde requirements als eerste opgepakt. RRBT volgt dezelfde gedachtegang: denken vanuit risico's en deze koppelen aan requirements.

RRBT heeft als basis dat op ieder moment van de testuitvoering de belangrijkste zaken zijn getest. Als door welke oorzaak dan ook de testduur beperkt is, bestaat zo een goed oordeel over de kwaliteit van de geteste

software op dat moment. Deze zienswijze sluit goed aan op de best practice 'iteratief ontwikkelen'.

Ook de best practice 'managen van requirements' en deze koppelen aan risico´s is de basis voor zowel RUP als RRBT. De software requirements zijn afgeleid van de wensen van de stakeholders en worden binnen RRBT gekoppeld aan risico's. Zo wordt verzekerd dat er geen risico's zijn zonder bijbehorende requirements. Het testen richt zich dan ook op het testen van die software requirements. De teststrategie bepaalt hierbij de prioriteit en de dekkingsgraad.

Past TestFrame bij RUP?
De software requirements die in de use cases en in de supplementary specifications staan, zijn de basis waarop wordt getest. Voor een testcluster binnen TestFrame kan vrijwel één op één een use case gebruikt worden. Dit bevordert de overzichtelijkheid en biedt de tester een handvat om testgevallen te herleiden.

Een testscenario wordt opgesteld met behulp van een businessscenario. Hierbij ligt de focus op de samenhang tussen verschillende use cases. De testclusters en testscenario's kunnen al opgesteld worden als de betreffende use cases opgeleverd zijn. Het systeem hoeft dus nog niet opgeleverd te zijn.

Omdat TestFrame een eenduidige en gestructureerde manier van werken en vastleggen afdwingt, wordt de herhaalbaarheid en overdraagbaarheid van de testsets gewaarborgd. Aan het eind van iedere iteratie worden de testclusters voor de bijbehorende use cases, of combinatie van use cases, opgeleverd. In alle volgende iteraties worden deze clusters opnieuw gebruikt om regressietesten uit te voeren. Deze manier van werken wordt ook nagestreefd volgens de best practices iteratief ontwikkelen, continue kwaliteitsverificatie en beheersing van veranderingen in de software.

Om de kwaliteitsontwikkeling goed te kunnen monitoren worden per cluster de testresultaten bijgehouden en in een vooraf gedefinieerde sheet verzameld voor rapportagedoeleinden. Het wordt daardoor eenvoudig inzicht te krijgen in de testresultaten en in de kwaliteitsontwikkeling (ook grafisch) van het systeem. Het is op deze manier mogelijk om de rapportage per cluster (use case) maar ook over het gehele testtraject te bekijken.

Het toepassen van TestFrame maakt het eenvoudig om over te stappen naar automatisch testen. De handmatige testset kan direct gebruikt worden als input voor de automatische test. Per project moet gekeken worden naar de haalbaarheid van automatisch testen. Als het aantal iteraties groot is en er binnen een iteratie ook herhaaldelijk wordt getest, is het interessant om (delen van) de test te automatiseren. De regressietesten kunnen dan bijvoorbeeld automatisch worden uitgevoerd.

Binnen TestFrame zijn diverse tools aanwezig die goed integreren met de Rational tooling.

Conclusie

Vanuit de best practices van RUP is gekeken naar RRBT en TestFrame. RRBT sluit aan bij het iteratief ontwikkelen en managen van requirements. TestFrame werkt ondersteunend bij de best practices iteratief ontwikkelen, continue kwaliteitsverificatie en beheersing van veranderingen in de software.

Daarnaast kenmerkt RUP kenmerkt zich door de risicogebaseerde aanpak. Deze aanpak sluit goed aan bij het risicogebaseerde testen van LogicaCMG. De op RRBT gebaseerde Testmanagement aanpak in combinatie met TestFrame is gestructureerd, eenvoudig, herhaalbaar en eenvoudig te automatiseren. Daarnaast is de aanpak completer en diepgaander dan de Testdiscipline binnen RUP.

Een belangrijk deel van de meerwaarde van het gebruik van TestFrame is dat de testanalyse meer in het proces betrokken wordt. Hierdoor kan gestart worden met testen voordat het systeem beschikbaar is.

Meer informatie over dit onderwerp is terug te vinden in het whitepaper 'RUP en TestFrame'. Daarnaast is er een training en een RUP plug-in 'RUP en TestFrame' gerealiseerd.

Deze en andere aanvullende informatie is op te vragen bij de auteurs van dit artikel.

(1) Een testscenario wordt opgesteld met behulp van een businessscenario. Hierbij ligt de focus op de samenhang tussen verschillende use cases.

(2) Testclusters zijn duidelijk afgebakende, onafhankelijke delen van een test. Testclusters bevatten onder meer testcondities en testgevallen.

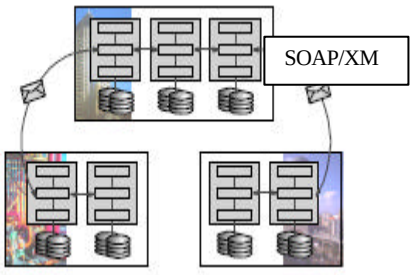
Testen van Service Oriented Architectures (soa)

Door Marcel van Donge/Jaap Wijnands
Donge@collis.nl

Om de concurrentiepositie te behouden of te versterken staan kostenreductie en productiviteitsverbetering bij organisaties hoog op de agenda. Daarnaast is het vereist om vaker en sneller te kunnen reageren op veranderingen in de marktvraag en technologie. Voor de toepassing van ICT betekent dit o.a. dat integratie van oude en nieuwe software eenvoudiger moet en dat er veel behoefte is aan platform- en technologieonafhankelijke software. Services Oriented Architecture (kortweg aangeduid als soa) is een veelbelovende architectuur die een raamwerk biedt om dit te realiseren. In dit artikel gaan we nader in op soa en op het testen van de software die in deze architecturen ontwikkeld en gebruikt wordt.

Soa verklaard

Een soa is een applicatiearchitectuur waarmee je op een gestandaardiseerde manier bestaande én nieuwe applicaties kunt integreren. Een soa bestaat uit stukken software die, gescheiden door goed gedefinieerde interfaces, elkaar als services aanroepen en waarmee software ondersteuning voor ketenintegratie mogelijk is. Als deze services via webtechnologie worden aangeboden noemen we dit ook wel webservices. Met een soa-architectuur wordt er gecommuniceerd over het netwerk, waardoor de services 'organisatie overschrijdend' aangeboden kunnen worden. Om een concreet voorbeeld te geven: in de reisbranche is het al redelijk gebruikelijk om op basis van deze architectuur reizen aan te bieden. Een consument kan kiezen tussen de aanbiedingen die de diverse reisorganisaties hebben. Op dezelfde manier komen bijvoorbeeld ook prijsvergelijkingen van allerhande producten tot stand. De data-uitwisseling vindt veelal plaats via XML (eXtensible Markup Language). SOAP (Simple Object Access Protocol) vormt daarbij de laag die bepaalt wat de service met deze data moet doen. Een SOAP bericht wordt meestal verstuurd met HTTP als transportlaag, maar soms is dat ook SMTP, HTTPS of FTP. Een en ander wordt verduidelijkt in onderstaande figuur.



Voorbeeld netwerk met soa-applicatie-architectuur

Soa zorgt voor homogeen opgebouwde applicaties en ICT-omgevingen. Omdat de keten servicegericht is kun je de keten eenvoudiger aanpassen, dit geldt zowel voor services als voor de bijbehorende afnemers. Hierdoor kan met behulp van een soa-architectuur de applicatie-integratie sneller, goedkoper en beter worden en minder risico met zich meebrengen.

Zo vroeg mogelijk testen

In de testwereld geldt de wetmatigheid dat hoe later fouten worden ontdekt des te kostbaarder het herstel er van is (Böhm). In architecturen als soa, waar (software)ketens geïntegreerd worden geldt dit des te sterker. Naarmate de ketenintegratie vordert wordt het des te moeilijker om van gevonden fouten de vinger op de zere plek in te keten te kunnen leggen. Dus, veel aandacht voor code review en unittesten. Gezien de aard van deze testen, die enigszins een herhalend en gelijkwaardig karakter hebben, kan testautomatisering een prima uitkomst bieden voor een hoge mate van efficiency.

Test de schakels, maar ook de keten

Gegeven de aard van soa (ketens) is er veel aandacht nodig voor het testen op de verschillende interfaceniveaus en de zogenaamde ketentesten. Met gefaseerd testen, één van onze Test Goal testprincipes, doe je in ieder geval altijd de juiste acties op het juiste moment. Zo beginnen we met het testen van de componenten, waarbij we veel met simulatiecomponenten als stubs en drivers werken. Daarnaast hebben we te maken met de zogenaamde translators voor het 'vertalen' van de berichten. Translators worden in een soa-architectuur gebruikt zodat applicaties goed geïntegreerd worden. Bij het testen van interfaces hebben we te maken met de opbouw van de interface maar ook met de effecten van wijzigingen, eigendomsverhoudingen en

storingsgevoeligheid. Omdat er op serviceniveau vaak geen user interface is worden er stubs en drivers ingezet om de protocollen (veelal in SOAP) en de XML berichten te testen. SOAP is de protocolstandaard voor de gehele keten en kan daardoor als één test uitgevoerd worden. Bij het testen van XML wordt het wat complexer, het gaat immers om meerdere soorten berichten.

Bij het testen van soa is het aspect ketentest belangrijk. De illustratie laat immers ook zien dat het werkproces, dat wordt ondersteund door het desbetreffende ICT-systeem, een keten kan zijn waarin meerdere webservices worden gebruikt. Het is belangrijk in welke volgorde de webservices worden uitgevoerd en onder welke voorwaarden. Bovendien is het denkbaar dat er binnen de ketens meerdere verzoeken tegelijk gestart worden met diverse webservices. Ook hiervoor zijn standaarden ontwikkeld en nog in ontwikkeling. Denk bijvoorbeeld aan UDDI (Universal Description Discovery and Integration; een standaard voor het beschrijven, vinden en gebruiken van webdiensten) en WSDL (Web Service Definition Language; een sleutelprotocol voor zakelijke webdiensten). Dit zijn standaarden die het mogelijk maken om de noodzakelijke volgorde van de webservices vast te stellen. Schaalbaarheid bij ketentesten is ook een belangrijk aspect; wat gebeurt er bijvoorbeeld als er een service bijkomt of vervalt. Hieraan is weer de performance gerelateerd; wat gebeurt hiermee bij wijzigingen in de keten. Als een consument bijvoorbeeld langer dan vijf seconden moet wachten op response haakt deze snel af.

Bruggen slaan

Een soa-architectuur betekent vrijwel altijd dat we met een keten van systemen te maken hebben. Om te controleren of deze keten de business-doelstelling realiseert is het belangrijk de juiste testen uit te voeren, waarbij we de werking van de keten goed moeten kennen. Een ander aspect bij een soa-architectuur is dat de businessbetrokkenen (bijvoorbeeld business-analisten en gebruikers) niet zoals gebruikelijk functionaliteit kunnen testen met behulp van een GUI.

Daaromdient de tester nog beter in staat te zijn om de brug te slaan tussen techniek en business. Dit maakt het verzorgen van de acceptatietesten van een soa-architectuur geheel anders dan bij meer 'klassieke' architecturen.

Security, zeker en vast

Security-testen is een belangrijk onderwerp. SOAP is een open systeem; de specificaties bevatten geen beschrijvingen van methoden en technieken voor de waarborging van authenticiteit, vertrouwelijkheid en integriteit van informatie. Door het ontbreken van deze waarborging zal SOAP voor de beveiliging van berichtenverkeer moeten vertrouwen op de beveiliging van het transportprotocol, bijvoorbeeld HTTPS of InfoMessaging. Dit betekent dat het testen hiervan hoog op de testagenda staat, zeker als de consument bijvoorbeeld zijn creditcard moet trekken.

Tenslotte

Soa is een veelbelovende architectuur die, met name door de voordelen die genoemd zijn in de inleiding, meer en meer gebruikt wordt. Als professioneel tester zul je deze architectuur waarschijnlijk steeds vaker tegen komen.

Service Oriented Test Architecture (SOTA)

Door Bart van Lunteren
bart.van.lunteren@logicacmg.com

Een vaste testaanpak in een ontwikkelomgeving met losse einden

Inleiding

Nieuwe methoden in softwareontwikkeling en architectuur vragen om een aangepaste testaanpak. Voorheen konden we op basis van uitgewerkte requirements, acceptatiecriteria, scherm lay-out en technisch ontwerp een goed beeld ontwikkelen van wat en hoe er getest moest worden.
Bij de flexibele ontwikkelingsmethoden zie je veelal dat de requirements parallel lopen aan de bouw van het systeem. De testvoorbereiding komt in de verdrukking omdat de ontwikkelcycli kort zijn. Een duidelijk voorbeeld van een dergelijke methode is Agile development, waar op basis van globale doelen en hieraan toegekende prioriteiten het ontwikkeltraject wordt gestart. Agile development is zo'n voorbeeld van evolutionair ontwikkelen. Het voortraject bestaat uit de afkadering van de bedrijfsprocessen die het informatiesysteem moet ondersteunen, er is een conceptarchitectuur en de

interfaces naar de te koppelen software services zijn bekend (z.g. 'point-to-point', 'hubs / brokers' integratie).
De verdere detaillering van de functionaliteit gebeurt in teams waarin alle expertise aanwezig is om een functionele eenheid te bouwen.
Passend in het Agile concept is SOA (Service Oriented Architectuur). SOA is een architectuurconcept met als belangrijkste kenmerken; een flexibel front-end (GUI) en een diversiteit aan programmatuur aan de back-end, die op afroep de gewenste services leveren. De verkeersagent (Enterprise Service Bus) zorgt vervolgens voor de end-to-end communicatie. Dit is een meer intelligente benadering van de traditionele point-to-point servicearchitectuur.

Testautomatisering en Agile development

De traditionele testautomatisering (record en playback of een iets meer geavanceerde vorm hiervan) is niet geschikt voor een flexibele ontwikkel omgeving. Dit omdat zo'n test ingeregeld is op de presentatie laag van de te testen applicatie. Dus moest er een aanpak komen waarmee de directe interface met de applicatie geëlimineerd wordt

SOTA

SOTA is een nieuwe ontwikkeling op testgebied. Het is een zogenaamde blackboxtest op basis van de TestFrame methode en de toolkit CASQCo (Complete Automated Software Quality Control) waarbij de test zelf volledig los staat van de Gui van de applicatie. Hetgeen je de vrijheid geeft om tests te ontwikkelen voor een gedefinieerde functionaliteit die volledig los staat van de presentatie van die functionaliteit.
De kern van de aanpak is dat je het te ontwikkelen informatiesysteem opsplitst in de min of meer vaststaande specificaties en dynamische delen. Stabiel zijn doorgaans de businessrules en de interfaces voor de te koppelen services. Dit zijn de elementen waarvoor in een vroeg stadium een testset ontwikkeld wordt. De dynamische delen zoals bediening, presentatie en controls worden dus buiten beschouwing gelaten. De volgende stap is dan dat je z.g. templates maakt waarin de structuur van de interface is vastgelegd. Uiteraard te beginnen bij die interfaces die uitgewerkt zijn. Meestal zijn dat de koppelingen met andere applicaties/modules. Zo'n template bestaat dan uit een vaste referentie naar de testdata en een format definitie. Als laatste kan een vertaaltabel nodig zijn om logische testdata om te zetten stuurcodes Voor het vastleggen van de testset, vertaaltabel en template(s) gebruiken wij Excel. Dit omdat je complexe interfaces inzichtelijk maakt en essentialia kan accentueren door het gebruik van kleuren.

De SOTA testaanpak is er op gericht om de test, los van de bediening/invoering interface- en los van de fasering van de oplevering, te ontwikkelen. In de uitvoeringstabel kun je aangeven welke (deel)test op welke service uitgevoerd moet worden. De uitvoering is dan alleen nog een druk op de knop.

De SOTA filosofie

De SOTA-benadering houdt in dat je een systeem als een aantal services beschouwt waarvoor je functionele tests definieert. De architectuur van de test(s) is gelijk aan de functionaliteit van de te testen applicatie. De testopzet is dus onafhankelijk van de uitwerking van die functionaliteit in de applicatie. Dit betekent dus dat de test ontwikkeling parallel loopt aan de ontwikkeling van de applicatie. Hoe de interface er uit gaat zien en dus hoe de testgegevens in het systeem ingevoerd worden is van later zorg en wordt opgelost met de interfacestructuur templates (loosely coupled principe). De inhoud van de interfacestructuur-template bestaat uit, de navigatie, de relatie met de testdata en een format-defintie van de testdata. Onderhoud ten gevolge van wijzigingen in de te testen applicatie is beperkt tot het template. Het testen van elke (deel) oplevering betekent een andere template aansturen, de testdata hiernaar converteren en uitvoeren op de applicatie.

Voorbeeld.

Stel dat een front-end module is ontwikkeld. Dan kan dat deel van de test uitgevoerd worden inclusief de controle op de interface van de nog niet beschikbare service en vice-versa.

SOTA Kenmerken.

- ontwikkelen van test(s) onafhankelijk van de volgorde van de opgeleverde functionaliteit;
- Vanaf specificatie en geautomatiseerd uitvoerbaar;
- Inzichtelijk;
- Interface onafhankelijk;
- Gering onderhoud;
- Incrementen van de functionaliteit van het systeem apart en integraal

testbaar met 1 test.

De testuitvoering bestaat uit de volgende stappen:

1. Inregelen welke (deel)test uitgevoerd moet worden;
2. Het geautomatiseerde testproces wordt gestart, waarin achtereenvolgens één of meerdere interfaces worden gegenereerd en direct ingevoerd in het te testen object. De gegenereerde test kan een interface zijn die losgelaten wordt op één of meerdere back-end services met controle op het resultaat. Dezelfde test kun je loslaten op de front-end GUI en het resultaat tegen het vanuit die test gegenereerde resultaat houden;
3. Als de interface een GUI betreft worden functies van het testtool aangeroepen.

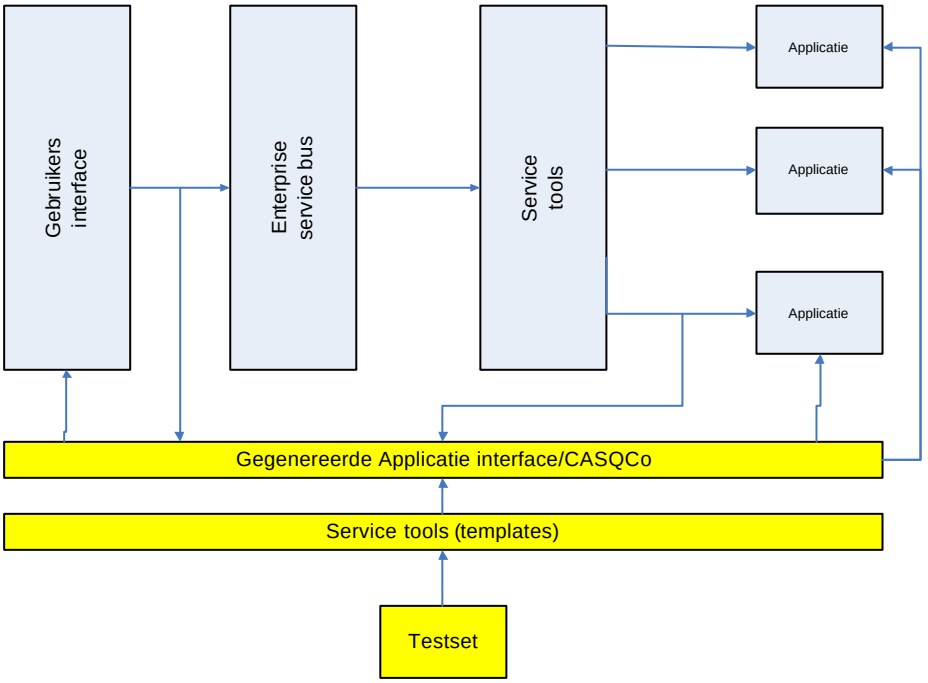
Het aardige is, dat met de test zowel de Gebruikersinterface, de interface naar de Enterprise Service Bus, de applicatie(= service) en de interface naar de applicatie, zowel als een geheel (front to end) als op onderdelen, getest kunnen worden aan de hand van de vanuit de testset gegenereerde (ijk)interfaces.
Als tester beschik je zo over een referentie voordat de software ter test wordt opgeleverd.

SOTA testware bestaat uit:

1. Testset(s);
2. Servicetools (templates)
   - Interfacestructuur-templates;
   - Vertaaltabellen om logische testdata te vertalen naar technische testdata (optioneel);
   - Programmatuur die zorgt voor de omzetting van de test(waarden) naar de interface template;
3. Gegenereerde applicatie-interfaces (CASQCo);
   - CASQCo: de toolkit die zorgt voor de invoering van de gegenereerde testdata in het systeem (front-end/back-end).

De praktijk.

Bij LogicaCMG HRM & Payroll Solutions kennen we drie salarisproducten, elk afgestemd op een marktsegment en daaromheen specifieke services. Voor het SOTA-pilotproject hebben we de materiedeskundigen van drie softwareproducten in een projectgroep bijeengebracht en hen de SOTA-architectuur uitgelegd.
De opdracht was, in het kader van de wet WALVIS, de loonaangiftetests te ontwikkelen. Daarnaast was het inrichten van een testomgeving, geschikt voor toekomstige SOTA, nodig. De test van de loonaangifte is gekozen omdat de functie nieuw is en functioneel 100%



Schematische weergave van SOTA
SOTA(geel), SOA(blauw)

gelijk voor de genoemde softwareproducten.
De boodschap is: focus op de overeenkomsten in functionaliteit, bepaal de servicecomponenten en hun interfaces en definieer standaards voor de interface templates, vertaaltabellen en de linking identificatieattributen.

De praktijk

De realisatie van de loonaangiftetest verliep niet synchroon voor de drie salarisproducten. Bijkomende problematiek was dat ook de belastingdienst steeds weer wijzigingen in de loonaangifte doorvoerde.
In juli 2006 zijn we begonnen met het inrichten van een SOTA-testomgeving. Dit betekent het flexibel aanspreken van diverse reeds bestaande testomgevingen vanuit één centrale SOTA-testwarebibliotheek.
Let wel, SOTA is een geautomatiseerde testomgeving waarbij gebruik wordt gemaakt van de bestaande testomgevingen van de te testen applicaties. De SOTA-testsets, interface templates en vertaaltabellen zijn ontwikkeld in eigen testomgeving.
Hiermee voldoen we aan de organisatorische voorwaarden van een centrale SOTA-testontwikkeling en -uitvoering voor overeenkomstige service en een decentrale testontwikkeling en testuitvoering voor specifieke testware.

De loonaangiftetest is een ophogende ontwikkeling die in juli is gestart op basis van de specificaties die op dat moment bekend waren, en is langzaam toegegroeid naar een volwaardige test. Voor de identificatie van de diverse rubrieken is gekozen voor de XML tags zoals de belastingdienst die voorschrijft. Binnen het project is één persoon verantwoordelijk gesteld voor het toezien op standaards. Voor de interface templates van de drie systemen hebben we de reeds bekende interfaces gebruikt en deze omgezet in Excel. Hierop hebben we de benodigde verwijzingen (XML tags) naar de rubrieken in de test aangebracht en een format-definitie toegevoegd. Achtereenvolgens zijn de vertaaltabellen per service ingericht en de interface templates gecompleteerd.

Zowel in de interface templates als de vertaaltabellen kun je zogenaamde default- of elimineercodes aangeven die, als er geen testwaarde voor is, worden ingevuld dan wel verwijderd uit de te genereren interface.
Dit was nodig om de asynchrone loonaangiftetest in de functionaliteitontwikkeling in de salarisproducten te overbruggen. Het grote voordeel was dat de testset ongemoeid bleef. En dan is er nog programmatuur ontwikkeld (40 uur) om de interfaces te genereren.

De testuitvoering

In de stuurtabel kun je aangeven op welke applicaties/services de test uitgevoerd moet worden. Daarna is het één druk op de knop om het proces te starten. Achtereenvolgens wordt de interface gegenereerd en uitgevoerd op de applicatie. Het resultaat van de test wordt tegen het referentieresultaat gehouden of, in geval van PayFact en PayMaster, tegen de loonaangifte als eindresultaat. Als we de test loslaten op PMS dan houden we de gemaakte PayFact-koppeling tegen de uit dezelfde testset gegenereerde PayFact-interface.

De ervaring

De initiële fase, het laten landen van de SOTA-architectuur, de testaanpak en het inrichten van vertaaltabellen, vertaalden zich in een inspanning die 100% hoger was dan de traditionele ontwikkeling van dezelfde testset voor één applicatie. Na de leer- en inrichtingsfase kon de extra inspanning van initieel 100% gereduceerd worden tot 30%. Hierdoor was de effectieve reductie per applicatie 35%. Deze reductie is toe te rekenen aan het wennen aan de SOTA-aanpak en het langzaam volwassen worden van templates en vertaaltabellen.
In het vervolg hierop zijn nog een aantal SOTA-tests ontwikkeld, allen gericht op nieuwe wet- en regelgeving voor 2006.

De tests zijn ontwikkeld voor:
- Bruto-/nettoberekeningen, 20.000 testgevallen uitgevoerd op drie applicaties;
- Voortschrijdend cumulatief rekenen uitgevoerd op twee applicaties;
- Auto van de zaak regeling uitgevoerd op twee applicaties;
- Ziektekostenverzekerings-wet uitgevoerd op twee applicaties;
- Levensloopregeling uitgevoerd op twee applicaties.

Over deze laatste tests is een effectieve reductie van 40% gerealiseerd. De echte grote winst was dat in het verleden de testset niet inzichtelijk was door de technische complexiteit van de interface, terwijl het nu

leesbare testsets zijn geworden waaruit dan weliswaar onbegrijpelijke interfaces worden gegenereerd.

Organisatie

Testsets waarmee meerdere systemen getest kunnen worden, zullen bij voorkeur opgehangen worden in een centrale testorganisatie of een project, waarin de gebundelde materiedeskundigheid beschikbaar is.

De succesfactoren bij het toepassen van SOTA zijn:
- Het definiëren en toepassen van duidelijke standaards;
- Het bundelen van materiedeskundigheid bij de testontwikkeling;
- Het centraal ontwikkelen en beheren van de testware;
- Afstemming met (decentraal) development is een must.

De voordelen zijn:
- Testontwikkeling voor oplevering van de betreffende functionaliteit;
- Geautomatiseerde testuitvoering in vroeg stadium;
- Niet afhankelijk van de fasering in oplevering van de software;
- Het gebruik van begrijpelijke, logische testwaarden, ook voor erg technische interfaces wat de inzichtelijkheid en overdraagbaarheid bevordert;
- De test, de interfacetemplates en vertaaltabellen in Excel. De toegevoegde waarde van Excel is dat je formules en betekenisvolle kleuren kan gebruiken;
- Een testset voor meerdere applicaties (overeenkomstige services);
- Applicatiemodules afzonderlijk maar ook integraal, zijn testbaar vanuit één testset;
- Tests kunnen in een vroeg stadium ontwikkeld worden en zijn in hoge mate ongevoelig voor wijzigingen in de programmatuur (weinig onderhoud);
- Materiekennis prevaleert over productkennis;
- Eenmaal ontwikkelen en N-maal gebruiken, levert een kostenbesparing variërend van 30% tot 50% (voor overeenkomstige services).

N.B: onderhoud bij wijzigingen in de services beperkt zich tot de interfacestructuur templates en de vertaaltabellen.

De nadelen zijn:
- Testers zullen zich het SOTA-concept eigen moeten maken en dat kost tijd;
- De initiële inspanning om SOTA in te richten is groot;
- Inrichting testomgeving (interfacetemplates en vertaaltabellen);
- Identificerende naamgeving testitems;
- Vertaaltabellen moeten worden ingericht;
- Complexiteit van de testuitvoering.

Conclusie

SOTA is een zeer hoopgevende ontwikkeling op testgebied, passend in het kader van flexibele ontwikkelingsmethoden- en technieken.
De toepasbaarheid van SOTA is vrijwel onbeperkt, wel moet steeds weer de afweging gemaakt worden of SOTA effectief is voor de te testen service. Expertise in SOTA moet ontwikkeld worden en tooling is een must.

Werkgroepen OPROEP

Door Bart Watertor
werkgroepen@testnet.org

Nieuwe initiatieven worden altijd gewaardeerd. Een bijdrage leveren aan Testnet is helemaal goed, en nog beter is het om een bijdrage te leveren aan de professionaliteit van testend Nederland. De werkgroepen zijn hiervoor een fantastisch middel. Momenteel hebben we een aantal succesvolle werkgroepen actief. Een aantal zullen zeer binnenkort hun resultaten met jullie gaan delen.
Dit succes willen we graag continueren, vandaar de oproep aan jullie om je aan te melden als werkgroeplid. Ook goede onderwerpen zijn van harte welkom.
Dus, heb je een goed idee voor een werkgroep en/of wil je je graag inzetten om een bijdrage te leveren aan de professionaliteit van jezelf en je vakbroeders? Mail naar werkgroepen@testnet.org.

Gezocht: Lustrum commissie

Door Michiel Vroon
evenementen@testnet.org

In 2007 bestaat TestNet 10 jaar en dat willen we niet onopgemerkt voorbij laten gaan. In een eerste brainstormsessie zijn verschillende mogelijkheden hiervoor in kaart gebracht. Voorbeelden van deze mogelijkheden zijn:
- een groots verjaardagsfeest; een uitgebreid evenement (eventueel op meerdere plekken in Nederland en met andere beroepsverenigingen);
- fotowedstrijd (“de testwerkplek”);

- of een special lustrumboek.

Voor het verder invullen van deze mogelijkheden zoekt de evenementencommissie uitbreiding. Draag je TestNet een warm hart toe en ben je ook van mening dat we dit lustrum niet zomaar voorbij kunnen laten gaan, meldt je dan aan bij Michiel Vroon via evenementen@testnet.org. Ook als je nog ideeën hebt kun je deze kwijt via dat bovengenoemde emailadres.

5 Vragen aan….

Door Pepijn van de Vorst, Ordina

-Ik vind testen een leuk vak want….

Testen is een veelzijdig vakgebied waar je met veel verschillende mensen te maken hebt en waar veel aspecten van een informatiesysteem bij elkaar komen. Met gebruikers en ontwerpers probeer je een beeld te krijgen van wat de applicatie zou moeten doen. Met bouwers en systeembeheerders probeer je een beeld te krijgen van hoe de applicatie zou moeten werken. Als tester vind je dan vaak verschillen tussen deze twee. Daarbij vind ik het leuk om uit te zoeken hoe een applicatie technisch en functioneel in elkaar zit en op basis daarvan de test zo effectief en efficiënt mogelijk uit te voeren. Daarbij komen uitdagingen om de hoek op het gebied van onduidelijkheden, eigenwijze ontwerpers, bouwers of systeembeheerders en communicatiestoornissen.

-Het grootste misverstand over testen is...

Dat het automatisch een stuk beter zou gaan als je methodisch gaat testen “volgens het boekje”. Als tester ben je voor een heel groot deel afhankelijk van de omgeving waarin je werkt. Daarbij zijn systematisch en gestructureerd werken, een goede verstandhouding met gebruikers, ontwerpers, bouwers en systeembeheerders en een diepgaand inzicht in de techniek en de functionele materie belangrijker dan het stipt volgen van een methode. Inzicht in de manier waarop het software ontwikkelproces werkt (niet op papier, maar in werkelijkheid) is belangrijk om invloed te kunnen uitoefenen op de uiteindelijke kwaliteit. Hiermee zeg ik overigens niet dat een methodische basis onbelangrijk is. Die is noodzakelijk maar niet voldoende.

-Over 5 jaar zie ik mijzelf in de functie van...

Adviseur op het gebied van verbeteren van software ontwikkelprocessen. Zoals dr. Edwards Deming al zei: “you can’t inspect quality into a product”. Uiteindelijk gaat het erom dat het proces van software ontwikkeling zodanig wordt ingericht dat er minder fouten gemaakt worden. Daarin zit de grootste kwaliteitswinst. Zover is het echter nog lang niet, dus voorlopig zal het testen een heel belangrijke rol hebben in de software ontwikkeling. Het testen kan daarbij een goede indicatie geven van de grootste struikelblokken in het ontwikkelproces.

Een tester moet zeker beschikken over de vaardigheid of kennis om...

Zich een eigen beeld te vormen van het doel en de werking van een applicatie. Een tester neemt weliswaar geen beslissingen m.b.t. ontwerp of bouw, maar vervult vaak wel een faciliterende rol in het duidelijk krijgen en helder communiceren van specificaties en functionaliteit. Een tester moet vriendelijk en toch beslist en vasthoudend kunnen zijn.

In de toekomst hoop ik dat binnen het test vakgebied … is veranderd, omdat…

Ik hoop dat testers en ook projectleiders / teamleiders zich meer bewust worden van het feit dat testen een schakel is in het ontwikkelproces. Om de kwaliteit van het eindproduct (software) te verhogen moet naar het hele proces worden gekeken. Testen is daarin belangrijk, maar niet zaligmakend.

Ik geef de vraag door aan…, omdat...

Ik geef de vraag door aan Peter Kalmijn, omdat ik hem ken als vakbekwaam tester en hij altijd een verfrissende en vaak verrassende kijk op de zaken heeft.

Known Errors

Er zijn 5 leden geweest die hebben gezien dat er juni 2004 in de header stond in plaats van 2006. Alert gereageerd, dank jullie wel. Ondanks onze

reviews glipt er toch nog wel eens iets tussendoor.
Verder is het voor de redactie ook heel leuk als we eens wat andere bevindingen toegestuurd krijgen. B.v. Leuk artikel, goede tip of mooie TNN.

Testtechnieken in de praktijk: De Elementaire Vergelijkingen Test (EVT)

Door: Rogier Ammerlaan
Rogier.ammerlaan@devoteam.nl

In een serie van korte artikeltjes in TestNet Nieuws vertel ik een aantal ervaringen die ik heb met verschillende testtechnieken, waarbij de theorie versus de praktijk vergeleken wordt.

In deze eerste, de elementaire vergelijkingen test (EVT). Deze techniek is bruikbaar voor onder andere functionele paden, verwerking in detail en systemen waar veel controles inzitten. EVT is volgens de theorie een grondige, zeer formele testtechniek en zeer arbeidsintensief.

Testbasis

Op basis van een functioneel ontwerp dat ik kreeg van een bedrijfskritische applicatie, welke een complexe vergelijking betrof, is besloten om EVT toe te passen.

Testanalyse

Als eerste is door een test analist het FO geanalyseerd en, zoals ook de eerste stap van EVT aangeeft, het FO vertaalt naar pseudo-code. Dit heeft geresulteerd in vier pagina's pseudo-code. Ik heb een klein voorbeeld opgenomen (blauw kader):

Review

Voordat we verder gaan met de testtechniek hebben we eerst deze pseudo-code uitgebreid met de functioneel ontwerper besproken. Al snel kwamen we er achter dat er fouten in het FO zitten. Deze kunnen we op deze manier snel corrigeren, zowel in het FO als in de pseudo-code. Dit heeft, in dit stadium, dus al 'vroege' en goedkope fouten opgeleverd. Fouten die anders gemaakt zouden kunnen worden tijdens bouw, waar de herstelkosten weer hoger zouden zijn.

Testgevallen (logisch)

Nadat de pseudo-code en het FO gecorrigeerd is, zijn we begonnen met het maken van testgevallen. Eerst per beslissing in de pseudo-code, hierna combineren en samenstellen van de logische testgevallen. Dit uiteindelijk heeft geresulteerd in 32 testgevallen. Even een vergelijking, met beslissingstabellen zouden we 1.073.741.824 (!) combinaties gehad hebben. Uiteraard heb ik hier niet even het aantal testgevallen uitgehaald dat niet zou kunnen.

Testgevallen (fysiek)

Hierna zijn de 32 testgevallen fysiek gemaakt en is de test uitgevoerd. Ieder testgeval heeft uiteindelijk wel geresulteerd in een of meerdere fouten in de programmatuur. Alle 32 testgevallen zijn uiteindelijk zoals verwacht door het systeem gegaan.

Aanvullend

Als extra aanvulling op deze formele testtechniek hebben we nog error guessing uitgevoerd door een aantal domein experts. Deze hebben ook nog een aantal zaken aan het licht gebracht die we niet gevonden hebben met de formele testtechniek EVT.

Een meting vanuit productie van deze applicatie heeft geleid tot een defect detection percentage van 98%. Er zijn wel bevindingen gevonden, maar dit zijn geen blokkerende zaken gebleken.

Wat wel lastig was bij deze techniek is dat tussentijds, terwijl logische testgevallen gemaakt worden, wijzigingen erg lastig te verwerken zijn in de structuur. Een stabiel FO is wel een vereiste. In ons geval is er een controle ergens midden in de pseudo-code bijgekomen. Om niet onze hele testspecificatie opnieuw te gaan aanpassen is besloten om deze testgevallen extra op te nemen naast de 32 bedachte. Het kost veel herwerk tijd om aanpassingen door te voeren in de testgevallen als er midden in de verwerking een controle toegevoegd wordt.

```
…
ALS kortingcode.expliciet_jn = 'n' DAN
        kc_id = NBW en kc_cd worden als input parameter meegegeven aan
        **** Toets aanvraag aan KRS Voorwaarden ****
        ALS kc_id (Laatste)is gevuld dan
                ALS kc_geldig = F (Als INGANGS_DTM>= Huidige_datum>= EIND_DTM) en expliciet_jn = 'J' DAN
                **** Foutafhandeling ****
                haal foutomschrijving 3 uit tabel foutsit
…
```

Conclusie

de techniek is inderdaad zeer arbeidsintensief en het kost veel tijd. Het levert wel een grondige en volledige test op, zeker gezien het hoge defect detection percentage. De testtechniek is inderdaad zeer zinvol gebleken voor deze complexe verwerking waarbij wel als randvoorwaarde gesteld moet worden dat de testbasis helder en stabiel moet zijn. Aanpassingen achteraf zijn lastig te verwerken. Aangevuld met een informele testtechniek zoals Error Guessing is deze techniek zeer dekkend gebleken.

TMap® Next, het nieuwe TMap® boek

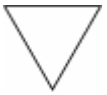
Door Tim Koomen, Bart Broekman, Leo van der Aalst en Michiel Vroon

Tien jaar na het eerste (Nederlandse) boek, vijf jaar na de tweede druk en drie jaar na de Engelse vertaling zijn we eind 2005 gestart met een grote vernieuwing van de methode TMap®. Op basis van ervaringen, ideeën en veel wijzigingsvoorstellen van testers over de hele wereld begonnen we met schrijven. Nu, 9 maanden later, kijken we terug op een effectieve periode met als resultaat een herziene en verrijkte methode. De eerste reacties van de externe reviewers zijn positief en geven ons het idee wat we ons werk goed hebben gedaan. Het plan is om het Engelse boek begin december te lanceren bij de EuroSTAR-conferentie (www.qualtechconferences.com). De publicatiedatum van het Nederlandse boek staat gepland op 20 december.

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

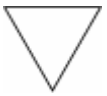
- De tijd heeft niet stil gestaan. Een groot aantal mensen vroeg daarom om actualisering van de methode, velen hebben hiervoor ideeën aangedragen;
- Dát testen nodig is, is in 2006 voor de meeste organisaties wel duidelijk, de discussie gaat veel meer over het hoe lang, hoe veel en wat getest moet worden;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

Het nieuwe TMap® vatten we samen in vier essenties:



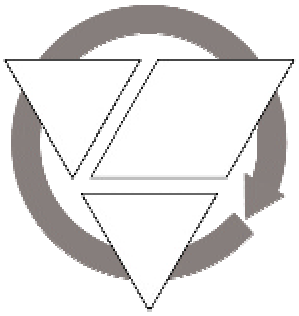
Het is gebaseerd op een business driven test management (BDTM) aanpak, die de opdrachtgever in staat stelt het testproces te sturen op rationele en economische gronden.

Het geeft een volledige beschrijving van het totale testproces.
TMap® bevat een complete 'gereedschapskist'. Deze gereedschapskist richt zich op de invulling van de onderwerpen technieken (hoe wordt er getest), infrastructuur (waar en waarmee wordt er getest) en organisatie (wie testen er).



Het is een adaptieve methode, flexibel toe te passen en daarmee geschikt voor alle testsituaties in de meeste ontwikkelomgevingen, zoals nieuwbouw, onderhoud, waterval / iteratief / agile ontwikkeling, maatwerk of pakketsoftware.

Deze essenties vormen samen onderstaand essentiemodel:



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

De verandering van TMap® heeft niet alleen consequenties voor het boek. Ook trainingen, certificeringen, websites enzovoort worden aangepast. Voor de laatste informatie hiervoor zie de website www.tmap.net

Overzicht boek

Bij het vernieuwen van de methode hebben we te maken gehad met een aantal uitdagingen. De vele wensen tot vernieuwing stonden soms haaks op het besef dat veel organisaties al naar volle tevredenheid werken volgens TMap® en niet erg enthousiast zijn om de werkwijze overhoop te gooien. We hebben er daarom voor gekozen de hoofdbestanddelen van TMap® intact te laten, maar in de details de benodigde vernieuwing te verwerken. Hierbij zijn we zo zorgvuldig mogelijk tewerk gegaan. Dit betekent voor de lezer een stuk herkenning. Met name de hoofdlijnen van de procesbeschrijvingen aan de hand van fasen, met per fase diverse activiteiten, en de aandacht voor technieken, organisatie en infrastructuur zijn blijven bestaan. Hier bovenop hebben we diverse vernieuwingen aangebracht. Verder is het gehele boek verrijkt met tips, uitdiepingen en praktijkvoorbeelden. Het boek is verdeeld in drie delen en deze staan hieronder beschreven.

Deel 1: Algemeen

Het eerste deel van het boek bevat een introductie op testen en TMap®. De hoe en het wat worden beschreven en de vier essenties van TMap® worden toegelicht.

Deel 2: Processen

Dit deel beschrijft de verschillende processen en hun onderlinge relatie die van belang zijn bij testen. Dit is inclusief ondersteunende processen. De beschreven processen zijn mastertestplanning, acceptatie testen, systeem testen en unit testen. Hierbij is speciaal aandacht geschonken aan de relatie tussen deze processen en het ontwikkelproces, BDTM en testmanagement. Naast de fasen Planning, Beheer, Voorbereiding, Specificatie, Uitvoering en Afronding is een nieuwe fase geïntroduceerd: Inrichting en beheer infrastructuur. In de fasen zelf is beter onderscheid gemaakt tussen de activiteiten van de tester en testmanager.

Naast deze primaire testprocessen worden ook ondersteunende processen beschreven. Dit zijn processen die gerelateerd zijn aan:

- Organisatie: het organiseren van een permanente testorganisatie, van Test Expertise Center tot Test Fabriek en het outsourcen van testen;
- Testomgevingen: eisen gesteld aan testomgevingen, processen voor het beheren van testomgevingen;
- Testtools: soorten testtools en hoe deze in te voeren;
- Testprofessionals: de functies, carrièrepad, karaktereigenschappen en trainingen.

Deel 3: Componenten

Het laatste deel van het boek bevat de zogenaamde componentenhoofdstukken. Hierin staan onderwerpen beschreven die in voorgaande processen op meerdere plekken gebruikt kunnen worden. Voorbeelden hiervan zijn:

- Productrisicoanalyse (PRA) De stappen voor het uitvoeren van een succesvolle PRA worden uitgelegd;
- Testvormen Verschillende testvormen (regressietest, usability test, performancetest, portabilliteitstest en securitytest) worden gedetailleerd beschreven;
- Begrotingstechnieken Verschillende technieken, inclusief de Test Punt Analyse, voor het begroten van de testinspanning worden beschreven;
- Testontwerptechnieken Als eerste worden de concepten van testontwerp en testdekking toegelicht. Hierna wordt een set van basistechnieken en een basisset van testontwerptechnieken beschreven.

Evenementen

SPI in Nederland: Resultaten, trends en uitdagingen voor de toekomst

PLAATS	ZEIST
GEBOUW	KNVB SPORT EN CONFERENTIE CENTRUM
DATUM	3 OKTOBER
TIJD	9.30 – 20.00

Belangrijk : Inschrijven via www.spiderconferentie.nl.
Testnet leden kunnen deelnemen aan de conferentie met een korting van EUR 25,-. De prijs wordt dan EUR 324,-. De kortingskode "Testnet" kan gebruikt worden bij aanmeldin g

Thema-avond "Testen van chips (in de financiële wereld"

PLAATS	NIEUWEGEIN
GEBOUW	: NBC
DATUM	24 OKTOBER
TIJD	18:00 - 22:00

Belangrijk :
Aanmelden uiterlijk 1 maart
E-mail: evenementen@testnet.org
Fax: 055 - 5415715

Thema-avond "Presentatie van de werkgroepen Metrics en Testautomatisering"

PLAATS	NIEUWEGEIN
GEBOUW	: NBC
DATUM	13 DECEMBER
TIJD	18:00 - 22:00

Belangrijk :
Aanmelden uiterlijk 1 maart
E-mail: evenementen@testnet.org
Fax: 055 - 5415715

Colofon

TESTNET BESTUUR

Bob van de Burgt	Voorzitter
Hans van Loenhoud	Vice-voorzitter & 2e penningmeester & Marktverkenning
Han Toan Lim	Penningmeester
Hans van Loenhoud	Secretaris & Ledenadministratie
Meile Posthuma	Informatievoorzien ing en beheer
Hans van Loenhoud a.i.	Marktverkenning Informatievoorzien ing & Beheer
Michiel Vroon	Evenementen & Thema-avonden

TESTNET MARKTVERKENNING, INFORMATIEVOORZIENING EN BEHEER
Hans van Loenhoud a.i. (T)

TESTNET WEB
Meile Posthuma (T)
Bob van de Burgt

TESTNET NIEUWS
Meile Posthuma (T)
Milo van der Kruis
Hein Baan
Johan Vink
E-mail: tnn@testnet.org

TESTNET EVENEMENT & THEMA
Michiel Vroon (T)
Rik Marselis
Cees Dulfer
Ine Lutterman-Baars
Bart Knaack,
Guido Dulos
E-mail: cie-ce@testnet.org (algemeen)
E-mail: evenementen@testnet.org (aanmelden)

TESTNET LID WORDEN
U kunt lid worden door een e-mail te sturen naar de ledenadministratie of door op onze Internet site het on-line registratieformulier in te vullen.
Internet site: www.testnet.org

TESTNET LEDENADMINISTRATIE
Hans van Loenhoud
E-mail: ledenadministratie@testnet.org

TESTNET NIEUWS©
TestNet Nieuws verschijnt eenmaal per kwartaal. Kopij aanleveren per e-mail aan de redactie
Het is niet toegestaan om de nieuwsbrief of delen eruit zonder bronvermelding over te nemen.

Legenda: (T) = Trekker aandachtsgebied