TESTNET NIEUWS

het online en interactieve testmagazine

Voorjaarsspecial 2014 - Testen – de praktijk!

TESTNET NIEUWS

Mei 2014 • Jaargang 18 • Voorjaarsspecial

www.testnet.org secretaris@testnet.org

VAN DE REDACTIE

Door Rob van Steenbergen • redactie@testnet.org @rvansteenbergen

En dan ben je opeens bezig met het schrijven van een inleidend stukje voor het blad dat je zelf altijd hebt gelezen, waarvoor je hebt geschreven en dat je nu helpt om te realiseren. Een blad en nog meer dan dat. We zijn ook online te bewonderen als échte testblog met artikelen vanuit de gehele TestNet bevolking - nieuws.testnet.org. En nog meer: er verschijnen artikelen van testers die geen lid zijn van TestNet. Meer dan een blad, een verlengstuk van TestNet, een platform voor kennisdeling voor testers. En ik ben er trots op dat ik hierbij mag helpen.

Degenen die nog niet het online blog volgen zijn wellicht verbaasd om in plaats van Hans van Loenhoud, iets van mijn hand te lezen. Vanaf begin dit jaar heb ik het stokje van Hans overgenomen. Gelukkig werkt Hans nog steeds mee als redacteur bij het maken van het TestNet Nieuws.

Een redactieteam dat je dit jaar nog meer gaat brengen. En een team dat aan je vraagt om hiermee te helpen. Heb je een idee wat je wilt delen met andere testers? Loop je misschien al langer met het verlangen om je ervaringen op testgebied te delen? Dan zou ik willen zeggen: twijfel niet langer en maak tijd door te beginnen met iets op te schrijven. Ga elke avond maar eens vijf minuten zitten om iets te schrijven. Langer mag ook, maar begin eens met vijf minuten.

Zo, dat was in ieder geval het pleidooi voor het delen van je ervaringen door middel van dit communicatiekanaal (mail me!). Nu rest mij eigenlijk niets meer dan je veel leesplezier te wensen met deze goed gevulde voorjaarsspecial. Ik hoop dat je hier veel ideeën uit kunt halen die je zelf in de praktijk kunt brengen! ←

COLOFON

Redactie
Paul Beving
Kees Blokland
Astrid Hodes
Hans van Loenhoud
Gerben de la Rambelje
Johan Vink
Rob van Steenbergen
redactie@testnet.org

Bestuur

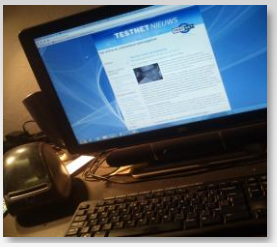
Rik Marselis	Voorzitter
John de Goei	Penningmeester
Peter van Tulder	Evenementen & thema-avonden
Kees Blokland	Informatievoorziening & beheer
Bernd Beersma	Marktverkenning & werkgroepen
Harro Philip	Secretaris & ledenadministratie

Opzeggen lidmaatschap: http://www.testnet.org/algemeen/algemene-voorwaarden.html#opzeggen

In dit nummer

Nieuws.testnet.org – TestNetNieuws wekelijks online

De TestNetNieuws 'Weekly' verschijnt iedere week in de vorm van één artikel op de website. Surf eens naar onze nieuwe TestNetNieuws op http://nieuws.testnet.org!



VAN DE VOORZITTER

Door Rik Marselis ● voorzitter@testnet.org

'Voor en door doeners', dat is een mooie typering van het doel van onze vereniging. Het is ook de ondertitel van het voorjaarsevenement 'Testen – de praktijk'.

Zoals jullie allemaal dagelijks merken, is het verschil tussen theorie en praktijk in theorie nooit zo groot als in de praktijk. Kortom, in de praktijk leer je pas hoe het echt werkt. Je bent nooit te oud om te leren, maar omdat je nooit oud genoeg wordt om alle fouten zelf te maken, moet je vooral leren van fouten van anderen. Die ervaring van anderen is door de redactie van de TestNet Nieuws verzameld en bij elkaar gebracht in dit mooie themanummer; ervaringen van jou en je vakgenoten.

Veel van deze verhalen komen ook aan bod tijdens het voorjaarsevenement op 14 mei 2014. Tijdens het evenement moet je kiezen tussen de lezingen die in de verschillende paralleltracks worden gehouden, in deze TNN kun je alle verhalen die je interesseren, lezen.

Waar gaat het heen met het testvak? Dat is een vraag die mij regelmatig bezighoudt. De tijdgeest verandert (zoals altijd); Agile en DevOps maken dat de rol van 'de tester' wezenlijk verandert.

Heb je na deze TNN nog behoefte aan meer leesvoer, dan kan ik je het boek 'The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win' van harte aanbevelen. Het is een roman over een IT-manager die met vallen en opstaan de samenwerking tussen IT'ers tot stand brengt en het IT-proces op orde.

IT is tegenwoordig zo complex dat het voor één persoon langzamerhand onmogelijk wordt om aan alle eisen te voldoen die aan de moderne tester worden gesteld. Dus wat moet je dan?

Juist: samenwerken. De kwaliteiten van verschillende mensen bundelen. En kennis uitwisselen. Dan kom je vanzelf bij onze mooie vereniging waarin de cumulatieve ervaring van ruim zestienhonderd testers beschikbaar is. Ik ben er trots op dat we al voor het zeventiende jaar deze bron van inspiratie voor elkaar zijn.

Oh ja, en als je je afvraagt: waarom schrijft Michiel Vroon deze opening niet? Na negen jaar in het bestuur waarvan vier als voorzitter vond hij het tijd voor 'vers bloed'. Tijdens de algemene leden vergadering op 13 maart jongstleden heeft Michiel afscheid genomen en hebben de leden ingestemd met de nieuwe bestuurssamenstelling waarbij ik de voorzittersrol heb mogen overnemen. Michiel, bedankt voor al jouw inzet in de afgelopen jaren!

Veel testplezier!

Rik Marselis

Voorzitter TestNet ←

TESTEN VAN EEN MOBIELE APP IN DE KETEN

Door Egbert Bouman ● egbertbouman@valori.nl @EgbertBouman

Mobiele Apps bieden verzekeraars nieuwe mogelijkheden om kosten te reduceren en tegelijkertijd de 'user experience' van de klant te verbeteren. Dit is een verhaal over de specifieke (test)uitdagingen van het inpassen van zo'n Mobiele App in de totale verwerkingsketen. De evaluatie van het project loopt momenteel nog en is vertrouwelijk. Een tipje van de sluier opgelicht …

Declareren is duur

Een belangrijk deel van de administratie - en dus van de kosten - van verzekeraars zit in het verwerken van door de verzekerden ingediende declaraties. De data wordt gecontroleerd, gevalideerd en doorgezet naar de database van het backoffice-systeem zoals OHI (Oracle Health Insurance), IDIT (een Israëlisch pakket), Level-7 (van de Nederlandse leverancier CCS) of een eigen maatwerkapplicatie. Elke maatregel die dat proces sneller en soepeler doet verlopen betekent kostenreductie en veelal ook snellere en betere service voor de klant.

Gebruikelijke maatregelen zijn:

- Uitbesteden van de declaratieverwerking aan een externe servicepartner;
- De klant alle gegevens laten invullen via een portaal, wat het werk naar de klant verplaatst;
- Scannen van de papieren declaraties gevolgd door veld- en karakterherkenning (OCR) en automatisch overzetten van de content naar het backoffice-systeem;
- Idem, maar dan op basis van scans die per e-mail of via een webportaal binnenkomen.

En Mobiele Apps op de smartphone van de klant bieden nu nieuwe mogelijkheden.

Straight Through Processing

Automatische verwerking van scans lukt anno nu nooit voor de volle honderd procent automatisch. Er is altijd een percentage uitval dat geheel of gedeeltelijk handmatig moet worden verwerkt. Het percentage dat wel geheel automatisch wordt verwerkt wordt het Straight Through Processing (STP) percentage genoemd. Hoe hoger het STP-percentage, hoe mooier en goedkoper!

Declareren met een Mobiele App: de ambities

Als je nadenkt over verdere optimalisatie van het declaratieproces dan ligt een Mobiele App voor de hand. Dat is dus precies wat men bij <grote verzekeraar> heeft gedaan. De belangrijkste ambities bij de start van dit project waren:

- PR: een App is hip en kan veel positieve exposure opleveren;
- Gemak voor de klant: wat is er makkelijker dan je nota's via een dedicated app even scannen met je smartphone en op een veilige manier direct naar je verzekeraar sluizen?
- Kostenreductie: door een hoger STP-percentage.

Voor de Denkers In Mogelijkheden (DIM'ers) was het niet zo moeilijk om hier een klinkende business case bij te definiëren. En dat was ook de basis voor het project 'Digitaal Declareren'. ➔

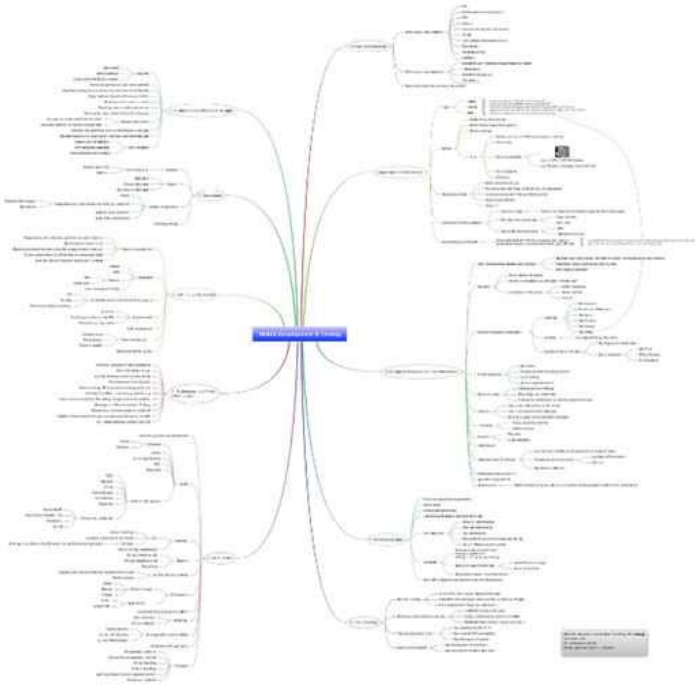
Van DIM'men naar DIP'pen

Tegelijkertijd is het niet moeilijk om de leeuwen en beren te vinden. Wie wil Denken In Problemen (DIP'pen) kan bovenstaand rijtje moeiteloos omturnen:

- Negatieve PR: met een App sta je in de etalage, inclusief mogelijke negatieve reviews in de App Store;
- Klagende klanten: als de App niet stabiel is op alle mobiele platforms, heb je een dissatisfier voor je klanten gecreëerd in plaats van een blijmaker;
- Hogere kosten: als de klanten hun smartphone niet goed stilhouden of met slecht licht scannen, dan is de kwaliteit van de scans belabberd en is het STP-percentage misschien wel veel lager dan bij gescande papieren.

Ketenrisico's!

We hebben uiteraard allerlei App-specifieke risico's geëvalueerd. Bijgaande mindmap / checklist, ontwikkeld door collega Derk-Jan de Grood, zou zeker waardevol zij geweest als die tijdig beschikbaar was geweest. De afbeelding is hier wellicht wat onleesbaar klein, maar geeft toch een goede indruk van de veelheid aan aspecten die van belang kunnen zijn.



Het plaatje is een weergave van een kwaliteitsmodel voor 'Mobile Development & Testing' als goed App-specifiek alternatief voor de ISO9126 / ISO 25010 kwaliteitsmodellen!

Maar … bijna alle serieuze discussies bleken een ketenaspect te hebben. De business stakeholders waren zuinig op hun STP-keten en zaten niet te wachten op een nieuw kanaal dat tot een stijging van additionele handmatige verwerking zou kunnen leiden.

Als een scan niet 'Straight Through' kan worden verwerkt, kun je dat dan terugkoppelen naar de klant en hem helpen een betere scan te maken? Als dat volautomatisch kan, dan is dat veel goedkoper dan de scan handmatig verwerken. Maar het vergt wel een complexe terugkoppellus, die zowel door de App als door de gehele keten moet worden ondersteund. Het leek geen goed idee … ➔

Eisen van de business stakeholders

De belangrijkste business stakeholder was degene die zich verantwoordelijk voelde voor het STP-percentage. Haar eis was in eerste instantie: de App moet 100% STP opleveren.

Dat heeft heel wat stof tot discussie opgeleverd, want is dat een reële eis? Nee, reëler leek de eis: Het STP-percentage van declaraties via de App moet minimaal op het niveau liggen van het STP-percentage van gescande papieren declaraties.

Je kunt zelfs nog een stapje verder gaan. We hebben de PR-voordelen en het extra gemak voor de klant. Bovendien vervalt de bewerkelijke stap van papieren inscannen dan wel inkomende mails verwerken. Dus zelfs met een iets lager STP-percentage is de business case nog gezond. Maar de stakeholder die dit brede perspectief met autoriteit inbracht, ontbrak in eerste instantie.

SMART requirements?

Een voorbeeld van een door de business bij het testteam neergelegde eis is onderstaande requirement, die ik hier maar even ongeredigeerd doorgeef:

De leesbaarheid van de declaraties via de App is gelijk aan de online keten:

a) 80,34% van de nota's krijgt kwalificatie 'goed'
b) 14,53% van de nota's krijgt kwalificatie 'redelijk'
c) 0,85% van de nota's krijgt maximaal kwalificatie 'matig'
d) 0,85% van de nota's krijgt maximaal kwalificatie 'slecht'
e) 3,42% maximaal van de nota's zijn 'leeg' (foute testgevallen, komen in productie niet voor)

Dit lijkt op het eerste gezicht heel mooi kwantitatief en meetbaar, maar als je wat langer kijkt komen de vragen:

- Waarom de huidige cijfers uit de online keten als maatstaf nemen?
- Is het reëel om de bestaande keten te vergelijken met een nieuw kanaal?
- De cijfers zijn blijkbaar meetwaarden, maar waarom twee cijfers achter de komma? Als requirement zou je rondere cijfers verwachten.
- Wat is de definitie van 'goed', 'redelijk', 'matig' en 'slecht'?
- Wat moeten we eigenlijk met categorie e. aan?

Wat kun je testen?

Het testteam werd onvermijdelijk deze discussies ingetrokken. Zij kregen de opdracht: toon aan dat het STP-percentage met de nieuwe App minimaal 90% is. Dat oogt als een redelijk 'business driven' testdoel, maar is dat ook zo?

Nou, nee dus! Je kunt de App testen, je kunt de keten testen, maar je kunt niet het gedrag van je klanten 'testen'. Daar kun je hooguit onderzoek naar doen, maar daarvoor voelde het testteam zich terecht niet verantwoordelijk.

Wat wel als een 'normale' testklus kon worden benaderd is dit:

- Je kunt je App op zichzelf technisch en functioneel prima testen: doet ie wat ie volgens het ontwerp moet doen? Kun je je profielgegevens goed invullen? Is een en ander voldoende beveiligd?
- Ook de communicatie met de STP-straat is prima te testen: komen de declaraties goed binnen en wordt een kwalitatief goede scan juist verwerkt? ➔

- Je kúnt ook de kwaliteit van je veld- en karakterherkenning, en de juiste mapping op de databasevelden testen. Dat vergt wel een standaard referentieset van scans. Die was er niet echt.

Deze en dergelijke zaken vielen binnen de normale kaders die men gewend was. Natuurlijk, er waren testbevindingen, discussies en issues. Maar allemaal binnen bekende kaders.

Wat kun je niet testen?

Veel lastiger waren de issues buiten de vertrouwde kaders:

- Je kunt niet de discipline van de klanten testen: houden ze hun smartphone recht boven de nota, zorgen ze voor goed licht, trillen ze niet? Daar kun je wel onderzoek naar doen uiteraard, maar is dat testen? Nee dus.
- Je kunt wél functionaliteit en ondersteunende functies in de App bouwen die de klant helpen om een betere scan te maken. En die functies kun je inderdaad testen.
- De kwaliteit van je scans is ook nog eens afhankelijk van de kwaliteit van de camera in je smartphone. Welke variaties in hardware en besturingssystemen moeten we allemaal meenemen? Die vraag werd rijkelijk laat gesteld en is nogal pragmatisch geadresseerd met de devices die 'toevallig' in de organisatie (projectteam, testteam en betrokkenen) beschikbaar waren.
- Kun je de kwaliteit van de camera's 'testen'? Ja dat kan: uiteraard zijn er methoden en technieken om de kwaliteit van de opnamen in een laboratoriumopstelling te beoordelen. Konden we dat in het kader van het project even doen? Nee, dat dan weer niet.

Deze en dergelijke vragen gaven aanleiding tot voortdurende discussie over de verantwoordelijkheid en de scope van het project en vooral van het testteam. Met telkens weer het gevaar van een patstelling die levensbedreigend was voor het projectresultaat.

Pragmatische aanpak

Hoe voorkomen we dat we in een patstelling blijven hangen? Gewoon door nuchtere vragen te stellen zoals: OK, misschien is het STP-percentage in eerste instantie niet wat je graag had gezien, maar is dat erg? Je hebt in elk geval andere voordelen en de eerste versie van de App zal niet de laatste versie zijn. Dat is dan weer het gemak van een Mobiele App: je kunt eenvoudig betere versies uitrollen zonder dat dat irritaties wekt bij de klant.

Om toch een gevoel van comfort te creëren rond de 'niet testbare' eisen is besloten om de principiële discussie te parkeren en een 'scan workshop' te organiseren.

Een set van ongeveer honderd (papieren) declaraties is vanuit de App gefotografeerd en verstuurd. In een Excel-sheet is bijgehouden wat de kwaliteit was van de verstuurde foto's zoals deze op de smartphone stonden.

De verstuurde foto's (declaraties) zijn vervolgens visueel beoordeeld als 'goed', 'redelijk', 'matig' of 'slecht'. De criteria hiervoor bleken niet honderd procent objectiveerbaar, maar vertrouwen in de uitvoerende personen haalde ook die kou uit de lucht.

Resultaat

De App is met succes uitgerold en de eerste recensies in de App store waren lovend. ➔

Kritische recensies zijn er inmiddels ook en het is zaak dat die in een volgende release worden opgepakt.

De STP-percentages vallen niet tegen en de discussie of de App wel zo'n goed idee was, speelt niet meer. De App is een vanzelfsprekend nieuw declaratiekanaal geworden.

Nog niet perfect, maar wel een aanwinst.

Samenvattend

Het testen van een Mobiele App in de keten is niet alleen een technische uitdaging. Ook hier is 'langszij komen' bij de stakeholders en transparante communicatie doorslaggevend.

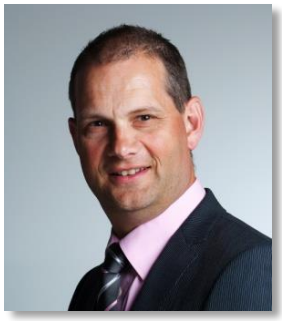
Het testteam koos er uiteindelijk voor om samen met de projectleider, de bouwers en de business stakeholders 'in de duivelsdriehoek' van tijd, geld en kwaliteit te gaan zitten.

De principiële standpunten werden geparkeerd en ingeruild voor de vraag 'wat kunnen we wél doen om de risico's voldoende te verkennen en minimaliseren'. Dat heeft geholpen en het resultaat mag er zijn.

(Met dank aan de stakeholders, projectleider en testers bij een grote verzekeraar) ←

PRAKTISCH USABILITY TESTEN VAN EEN MOLECULAIR DIAGNOSTISCH SYSTEEM

Albert Door Patrick Duisters ● patrick.duisters@improvegs.nl @PatrickDuisters



In dit artikel worden ervaringen gedeeld over het usability testen van een nieuw en innovatief (medisch) moleculair diagnostisch systeem in een snelgroeiend Belgisch bedrijf.

Op basis van een praktijkvoorbeeld beschrijf ik hoe gebruikte heuristieken, methoden en tools helpen bij de uitdaging om een transparant en controleerbaar 'usability programma' op te zetten in een multidisciplinair team van biologen, researchers en hard- en softwareontwikkelaars.

Omdat in het testdomein voor gebruiksvriendelijkheid veelal de term usability wordt gebruikt, zal deze term in het vervolg worden gehanteerd.

Het systeem

Het systeem bestaat uit een console, een of meerdere instrumenten en cartridges met daarin specifieke chemicaliën. Een patiënt-monster wordt met een pincet, pipet of swab in de cartridge gebracht. Na het sluiten van de gevulde cartridge is deze hermetisch gesloten en wordt ingebracht in een beschikbaar instrument, verwerkt en de data geanalyseerd. De console ondersteunt de workflow, zoals het aanmaken van een onderzoeksverzoek en het beschikbaar maken van het onderzoeksresultaat.

Het grote voordeel van het systeem is dat, met per test een specifieke cartridge, het onderzoek uitgevoerd kan worden met minimale gebruikershandelingen: 'Sample in – result out', in circa 1,5 uur. Zulke onderzoeken zijn nu arbeidsintensief en complex en worden daarom vaak in batches in centrale laboratoria uitgevoerd. Dit met wachttijden en onzekerheid voor de patiënt tot gevolg.

Risico management & Regelgeving

Toezichthoudende instanties zoals FDA vinden aantoonbare bewijsvoering, 'userfriendliness', 'learnability' en 'error prevention' belangrijk. Ook daarom is het in deze context belangrijk om te voldoen aan de ISO-62366 standaard voor 'usability engineering' in het medisch hulpmiddelendomein. Indien je aan deze standaard voldoet, hoef je aan de FDA minder uit te leggen.

Vanaf het begin van de ontwikkeling van het systeem is er al aandacht besteed aan usability, maar er werd nog gezocht naar een bij deze standaard passende 'usability' testaanpak. Daarom heb ik, als Test Architect voor Verificatie, hiervoor een testaanpak geschreven gebaseerd op deze standaard en de uitvoering daarvan begeleid.

Belangrijke elementen uit ISO-62366:

- De engineeringaanpak behorende bij risicomanagement, noodzakelijk voor medische systemen. Denk hierbij aan het voorkomen van (gebruikers)fouten zoals door invoercontroles een robuuste 'workflow', of het maar op één manier kunnen plaatsen van de cartridge in het instrument.
- Iteratief ontwikkelen en testen. Zo vroeg mogelijk en herhaald evalueren door middel van reviews en testen tijdens de ontwerp-, verificatie- en validatiefasen van de productontwikkeling (V-model).

Usability Programma

Op basis van het voorgaande is de volgende aanpak geadviseerd: ➔

1. Analyseer alle requirements en markeer die welke usability gerelateerd zijn. Daarmee wordt aantoonbaar dat usability gedurende de hele ontwikkeling een aandachtspunt is geweest.
2. Stel een usability testprogramma op met passende testen voor de verschillende V-model fasen (ontwikkeling, verificatie en validatie).
   Maak daarbij gebruik van bekende heuristieken en methoden, bekend uit de softwarewereld, zoals de Heuristic Evaluation [Nielsen, http://www.nngroup.com/articles/ten-usability-heuristics/]. en de Software Usability Measurement Inventory (SUMI) [http://sumi.ucc.ie/].

Het testprogramma bestond uit:

- Reviews en testen met interne gebruikers tijdens de ontwikkelfase.
- Testen en een enquête met externe gebruikers in het eigen laboratorium in de verificatiefase.
- Observaties en een enquête met externe gebruikers op hun werkplek tijdens de (klinische) validatie.

De uitvoering

De analyse van de requirements was een relatief makkelijke stap: alle systeemrequirements nalopen en markeren als er een usability aspect herkenbaar was. Gedurende de reguliere verificatietesten worden deze requirements geverifieerd en is het voldoen aan de usability aspecten bewezen.

Voor de uitvoering van het usability testprogramma hebben we een team samengesteld met een projectleider, een vertegenwoordigster van de biologisch laboranten en ikzelf als Test Architect.

Eerder opgestelde 'use cases' vormden een belangrijke basis voor de usability testen. Omdat deze nog niet compleet waren, hebben we deze eerst aangepakt. In een workshop zijn de fasen in de levenscyclus van het product vastgesteld: van installatie, via gebruik en service, tot en met verwijdering uit het laboratorium. Vervolgens hebben we per fase de use cases en activiteiten vastgesteld, en voorzien van eigenschappen zoals het type gebruiker, kennisniveau, en frequentie van gebruik.

De testen in de ontwikkelfase zijn exploratief aangepakt. Daartoe zijn er (exploratory testing) charters opgesteld voor enkele veelgebruikte use cases en voor onderwerpen waarop snel feedback nodig was, zoals het inbrengen en afkorten van een swab (een soort wattenstaafje) in de cartridge en het plaatsen van de cartridge in een instrument. In deze exploratory testing sessies was er vrijheid in uitvoering. In een paar uurtjes hebben we veel feedback verzameld. Als bewijs, 'for the record', zijn notities en video-opnames gemaakt.

Deze resultaten hebben het usability testprogramma herkenbaar 'op de kaart' gezet voor ontwikkelaars en het management. Daarmee werd ook de samenwerking tussen ontwikkelaars en testers bevorderd, alsook de verdere ontwikkeling van het systeem.

Reviews

Naast de testen zijn er usability reviews uitgevoerd in de ontwikkelfase. Hiervoor is de Heuristic Evaluation op basis van de tien heuristieken van Nielsen toegepast. Ondanks zorgvuldige voorbereiding en een kick-off met toelichting kwam deze review traag op gang. Een veelgehoord argument was: 'maar we hebben toch al gereviewd?'. Dat was waar, maar nooit met een focus op usability aspecten. Na herhaalde toelichting van het nut en toepassing van de heuristieken en de andere usability-resultaten werd met vertraging de gevraagde feedback verkregen. Diverse review-bevindingen werden ondertussen ook in verificatietesten gevonden. Een gemiste kans dus om ze ➔

vroegtijdiger te vinden en op te lossen. Een goede reden om hier bij een volgend project sterker in te zetten op tijdig (heuristisch) reviewen.

Testspecificaties

In de verificatiefase zijn er op basis van de use cases testscenario's opgesteld, gegroepeerd per gebruikersgroep (onervaren gebruikers, ervaren gebruikers, beheerders) en rekening houdend met de frequentie van gebruik. Deze zijn vervolgens door de laborante verder uitgewerkt tot testprotocollen. Deze protocollen hebben een beperkt detailniveau zodat nagegaan kon worden of de gebruikers zonder aanvullende toelichting zelfstandig met het systeem aan de slag kunnen. Zodoende kan de 'learnability' op de proef worden gesteld.

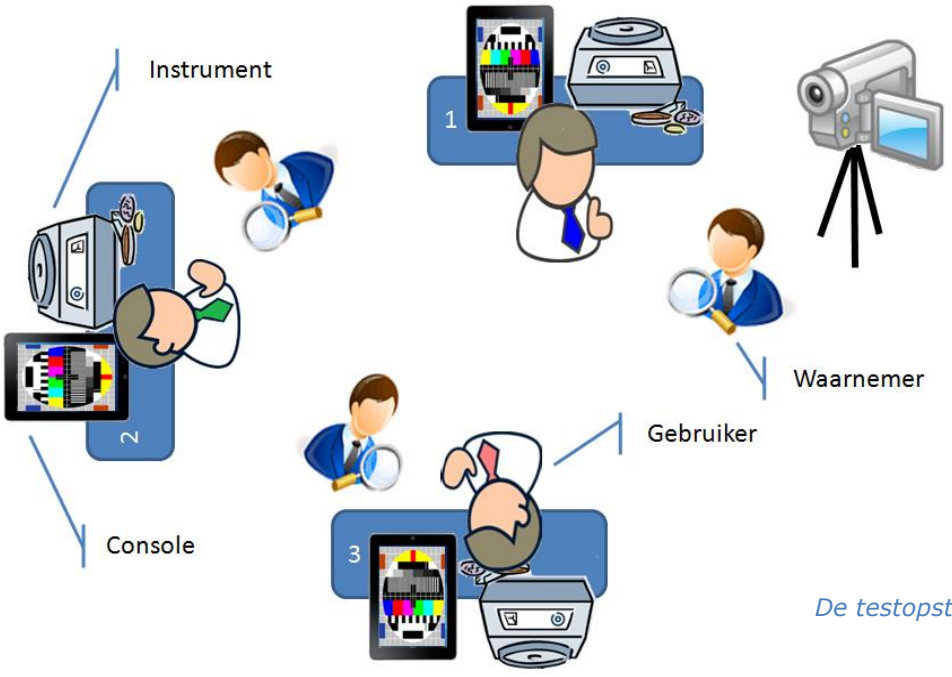
Testsessies

De volgende belangrijke stap was het vinden en uitnodigen van (externe) representatieve gebruikers voor de testsessies. Samen met de marketingafdeling zijn gebruikers gevonden met verschillende kennis- en ervaringsniveaus, een diversiteit aan leeftijden en afkomstig uit verschillende landen: van Noorwegen tot Gibraltar.

Per kennis- en ervaringsniveau zijn sessies georganiseerd. Na een algemene introductie van het bedrijf, het systeem en een toelichting over de usability-testen, kregen onze gasten een korte gebruikerstraining van circa dertig minuten bij het systeem in een laboratoriumomgeving. Daarna werden ze losgelaten met de testprotocollen. Per gebruiker (maximaal drie per sessie) was er één observator aanwezig om de handelingen van de gebruiker gade te slaan en notities te maken. Voor bewijsvoering en voor nadere analyse zijn er video-opnames gemaakt met een zichtbare strategisch opgestelde handcamera.

Hier kwam alles samen: de zorgvuldige voorbereidingen en het gebruiksvriendelijke systeem, met enthousiaste gebruikers als gevolg! De detaillering van de testprotocollen was precies goed. De gebruikers waren in staat om met de minimale instructie hun testtaken zelfstandig en conform planning uit te voeren.

Na afloop van elke testsessie is er een feedbackronde gehouden met onze gasten. Deze begon met het invullen van



De testopstelling

een enquête. De vragenlijst is gebaseerd op de SUMI questionnaire, aangevuld met soortgelijke vragen over het ➔

inbrengen van het patiëntmonster tot en met het beoordelen van de onderzoeksresultaten. Ook hebben we gevraagd of de gebruiker het systeem zou aanbevelen (Net Promotor Score) en naar hun 'Top 3' plus- en minpunten.

Al tijdens de eerste workshop (use cases) ontstond er een 'klik' in het usability-testteam met een zeer prettige samenwerking tot gevolg. Door de goede resultaten werd het usability-team nog enthousiaster. We kregen niet alleen positieve feedback van de gebruikers op het systeem, maar ook positieve feedback op het proces, van ontwikkelaars, management en gebruikers: 'One of the best usability sessions ever' en 'you have raised the bar'.

Subjectief versus Objectief

Een aspect van 'usability' is de subjectiviteit, die niet objectief te maken valt. Volgens [ISO 9126]: The capability of the software to be understood learned, used and attractive to the user when used under specified conditions. Regelmatig heeft het usability-testteam met de ontwikkelaars gediscussieerd over subjectieve onderwerpen zoals 'aantrekkelijk' en 'onhandig'. De analyse van observaties en enquêtes heeft de subjectieve meningen kwantitatief gemaakt. Meningen werden bevestigd, maar ook hebben we van de gebruikers geleerd over de gang van zaken in laboratoria in en buiten België, waarmee de researchers, ontwikkelaars en management hun voordeel kunnen doen.

Een test is niet compleet zonder …

… verwacht resultaat. Voor de requirements met het usability-aspect was de beoordeling van het verwachte resultaat eenvoudig. Hiervoor hebben we een inspectie uitgevoerd op de verificatie-testrapporten om te controleren of de testen uitgevoerd en de resultaten volgens verwachting waren.

Het requirement voor de 'learnability' was opgesteld in de trant van '90% van de gebruikers moet het systeem kunnen gebruiken na een beknopte training'. Op basis van de training van dertig minuten konden al onze gasten zonder problemen of hulp de testprotocollen uitvoeren. Mede op basis van de resultaten uit de enquête konden we vaststellen dat dit requirement ´pass´ was.

Het antwoord op de vraag of het 'attractive' is om met het systeem te werken, is afgeleid uit de enquêtes. Met een score van een dikke 8 was ook dit requirement 'pass'. De geïdentificeerde verbeterpunten worden opgepakt.

Conclusie

Engineering (ontwikkeling en testen) van 'non-functionals' is subjectief en lastig en wordt niet altijd als vanzelf opgepakt. Gebruik van methoden en heuristieken uit de softwarewereld vormen een goede basis en onderbouwing voor het opzetten van usability-testen, ook in een gereguleerde omgeving.

Met een enthousiast team heb je niet veel middelen nodig om heel mooie resultaten te bereiken. Bovenal heb ik veel geleerd en ben ik nog meer geïnteresseerd geraakt in usability en de menselijke kenmerken en eigenschappen van gebruikers. ←

TESTAUTOMATISERING MET ONTWIKKELTAAL IN EEN AGILE OMGEVING

Door Albert Eikelenboom ● albert.eikelenboom@cgi.com

De laatste jaren ben ik ingezet bij een klant als testautomatiseerder waarbij de klant diverse applicaties en systemen maakt met behulp van microsoft C# en Windows Presentation Foundation. In 2011 is men bij het ontwikkelen van applicaties overgestapt van waterval op een Agile aanpak, waarbij er sterk is ingezet op testautomatisering, qua tijd en budget. Voor mij betekende dit ook een overstap naar een andere ontwikkelomgeving, van VB scripts naar het programmeren in C#. Hoe is dit project opgezet en hoe is testautomatisering hier ingezet?

Het project

Het project wordt uitgevoerd met een aantal Scrumteams. Elk Scrum-team bestaat uit een Scrum-master, een producteigenaar, een aantal ontwikkelaars en een vaste tester. De testautomatiseerders zitten niet vast in een team. Om het Scrumproces beter te kunnen managen, komen de Scrum-masters elke maandagochtend bij elkaar voor de Scrum of Scrums.

Elk Scrum-team heeft een eigen softwarecode branch om de software te kunnen beheren. Er is een master software code branch, waar de teams met enige regelmaat naartoe kunnen integreren. Om de kwaliteit te bewaken zijn er een aantal quality gates, waaronder de automatisch uitgevoerde testen, waaraan voldaan moet worden voordat een team gaat integreren.

Het team testautomatisering

Het team bestaat uit zeven personen. Bijna alle teamleden zijn ervaren C# ontwikkelaars. Om ervoor te zorgen dat er grip blijft over de backlog werken zij met takenlijsten en werkafspraken. Verder is er een eigen testframework ontwikkeld voor de specifieke te testen applicatie. Het testframework controleert of er een nieuwe build beschikbaar is, die vervolgens getest wordt op een virtuele omgeving van vijf machines, waarbij er één de controller is en de andere vier de software installeren en testen (de agents). Er kunnen op verschillende omgevingen tegelijkertijd software builds worden getest. Het technisch beheer van de virtuele omgeving ligt bij ICT-beheer. Microsoft testmanager wordt gebruikt voor het borgen van de testgevallen en de resultaten.

De eenduidigheid van werken volgens het testframework is heel belangrijk, evenals het snel kunnen overpakken van elkaars werk. Hierdoor is er gekozen om als team van testautomatiseerders bij elkaar te zitten. Vanuit het team wordt capaciteit toegewezen aan de Scrum-teams.

Tijdens het ontwikkeltraject werkt het Scrum-team aan nieuwe software, die aan verandering onderhevig is. Tijdens dit proces krijgen de teams input over hoe de software gemaakt moet worden, zodat testautomatisering er gemakkelijk op kan inhaken.

In overleg met het Scrum-team worden afspraken gemaakt welke automatiseringstaken er in een sprint worden opgepakt. De software die in de vorige sprint is opgeleverd, is daarvoor het meest geschikt om op te pakken. Hierdoor ontstaat er in de loop van het ontwikkeltraject een regressietestset die dagelijks op de branches geautomatiseerd wordt uitgevoerd. ➔

Testautomatisering in ontwikkeltaal

Bij reguliere testautomatiseringsprojecten worden vaak tools gebruikt die Grafische User Interface (GUI) objecten kunnen herkennen waarmee je kan scripten. Tijdens Agile/Scrumprojecten worden regelmatig aanpassingen gedaan aan de GUI. Met als gevolg dat je de scripts moet aanpassen.

Het is een groot voordeel dat de ontwikkeltaal van de testtool en de applicatie hetzelfde is. Doordat de testautomatiseerders van hetzelfde ontwikkelniveau zijn als de applicatie ontwikkelaars kan je:

- de testcode inprikken onder de GUI laag. Deze laag zal minder vaak wijzigen, waardoor de testcode stabieler wordt.
- de testcode heel dicht tegen de ontwikkelcode aanzetten. Als de ontwikkelaars nu een API wijzigen, dan compileert de testcode niet meer, waardoor dit gelijk kan of moet worden aangepast.
- als een test faalt, makkelijk even in de code kijken om te achterhalen of het probleem in de testcode of in de ontwikkelcode zit.

Buiten bovenstaande voordelen heeft een testautomatiseringsteam dat in dezelfde ontwikkeltaal werkt als de applicatieontwikkelaars nog andere voordelen. Het team is in staat dashboards te maken die worden gebruikt voor de quality gates van het beoordelen van de softwarekwaliteit.

Bijvoorbeeld:

- de code coverage van de unit testen en van de automatische testen meten en weergeven.
- het maken van benchmarks, inclusief een website. Voor cruciale API calls, wordt er bijvoorbeeld drie minuten gemeten hoe vaak een bepaalde API call kan worden uitgevoerd. Dit wordt bijgehouden in een database en op een scherm. De veranderingen worden scherp in de gaten gehouden.
- het bijhouden van de code warnings per Scrum-team. Code warnings kan je in C# opdelen per namespace en per code warning. Op een dashboard krijgen de teams voor hun code een smiley. Het team krijgt een groene smiley, als ze bij alle namespaces en code warnings gelijk of minder warnings hebben dan de integrate build. Er is een rode smiley te zien als dat niet zo is. ➔

Feature Team Builds

Build	Latest Tested Build	Warnings	Suppressions	NameSpace Errors	Compared to Integrate Build	Warnings	Suppressions	NameSpace Errors	Status (Clickable)	Suppressed (Clickable)	Namespace (Clickable)
ELIAS_FT_EK_P	ELIAS_FT_EK_P_20140319.4	1	1 (0)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FT2_F	ELIAS_FT_FT2_F_20140319.3	2673	580 (260)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FT4_F	ELIAS_FT_FT4_F_20140319.2	2678	592 (272)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FT5_F	ELIAS_FT_FT5_F_20140317.2	2674	592 (272)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FTD_F	ELIAS_FT_FTD_F_20140319.1	2675	580 (260)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FTS_F	ELIAS_FT_FTS_F_20140317.1	2674	592 (272)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_FTST_F	ELIAS_FT_FTST_F_20140319.2	2676	580 (260)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_HYP_F	ELIAS_FT_HYP_F_20140318.1	2582	595 (265)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_INFRA_F	ELIAS_FT_INFRA_F_20140314.1	2674	592 (272)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_LRC_2014_F	ELIAS_FT_LRC_2014_F_20140314.1	2674	592 (272)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_FT_MNT_F	ELIAS_FT_MNT_F_20140319.1	2675	580 (260)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			
ELIAS_SETUP_F	ELIAS_SETUP_F_20140318.3	2671	586 (262)	0	ELIAS_INT_F_20140219.1	2674	598 (274)	0			

By TestAutomation

Figuur 1: Smiley voor code warnings, code onderdrukkingen en namespace errors per team

- Het maken van een dashboard voor de automatische testresultaten van de automatisch uitgevoerde tests. Als er vijf Scrum-teams zijn met elk hun eigen branch waarop 300 testen geautomatiseerd worden uitgevoerd, komen er 1500 testresultaten beschikbaar. De resultaten worden weergegeven met kleurcoderingen.

Geslaagde testen zijn groen. Gefaalde testen met een bekende bug zijn rood. Gefaalde testen zonder bekende bug zijn paars. Op deze manier weet het testautomatiseringteam dat ze de paarse testen moeten analyseren.

Verder wordt er ook gelogd op runtime errors en warnings. Als deze optreden, dan verandert tijdens een test ook de kleur van het testresultaat. Deze test wordt geel-oranje als er een bekende runtime warning of error optreedt, terwijl hij roze wordt als er een onbekende code warning optreedt. Zodoende weten de testautomatiseerders welke warnings nog optreden en welke nieuwe ze moeten melden en analyseren.

Figuur 2: Een dashboard van de automatische testresultaten

Conclusies

- Testautomatisering is een belangrijke factor bij Agile/Scrum ontwikkeltrajecten.
- Testautomatiseerders zullen behalve een scripttaal in de toekomst ook een echte programmeertaal moeten gaan leren.
- Door testautomatisering goed in te bedden in de Agile organisatie ontstaat er een kwalitatief hoogwaardige regressietestset die op verschillende software branches tegelijkertijd kan worden getest.
- Door goede en heldere dashboards te gebruiken kunnen veel testresultaten in korte tijd worden beoordeeld, waardoor de kwaliteit van de software goed kan worden bewaakt.
- De ontwikkelde quality gates, bijvoorbeeld voor aantal code warnings, de benchmarks, namespace errors, etcetera zorgen ervoor dat het (test) management goede tools in handen heeft om het softwareontwikkelproces onder controle te houden en de code te laten herstellen en te verbeteren. ←

TEMPLATES ZIJN ZO 2010

Door Maurice Siteur ● maurice.siteur@capgemini.com

Als ik dit soort opmerkingen hoor, dan moet ik altijd lachen. Mijn referentiekader van de IT strekt zich uit tot 1981, toen ik mijn eerste Basic programma moest maken en dat ronduit lastig vond. Dan is 2010 gisteren en beschouw ik dat soort opmerkingen als uitingen van onbezonnen jeugdig enthousiasme.

Het kan niet zo zijn dat nieuwe manieren van werken de oude manieren totaal overbodig maken. Dat is nog nooit gebeurd en dat zal ook niet gebeuren, hoe hard de Agile / context driven / mobile / cloud community dat soms ook roept.

Niet weggooien

Dat waterval had ook wel iets en dat moeten we niet weggooien. Dat is net als het kind met het badwater wegspoelen. Daar krijg je binnen no time spijt van. Dat wil niet zeggen dat we van de nieuwe wereld niet een hoop kunnen leren; iets wat voor sommige mensen dan weer heel erg moeilijk is.

Ik vraag me geregeld af waarom ik dat master testplan toch weer dertig pagina's dik heb weten te krijgen. Staat daar niet te veel in? Staan er wel genoeg plaatjes in? Kan ik bepaalde inhoud via tabellen beter leesbaar maken?

Maar dan denk ik toch elke keer weer dat organisaties die niet een gestroomlijnde organisatie hebben, waarin testen niet volledig geïntegreerd is in het ontwikkelproces wel graag een totaalbeeld willen hebben van wat er geregeld moet zijn. En dan nog heb je het acceptatieproces dat over verschillende schijven loopt: de afspraken daarover moet je ook vastleggen. Of misschien is het wel voldoende om alleen de acceptatiecriteria vast te leggen? ➔

Templates

In een geoliede machine passen templates en vaste werkwijzen. Templates zijn een vastgestelde manier waarop zaken worden vastgelegd. Doordat je templates gebruikt, hoef je niet meer na te denken over de vorm en de opzet. Die is er en die is leidend voor wat er moet worden opgeschreven.

Dit geldt ook voor documentatie. Daarmee kunnen nieuwe teamleden naar het verleden kijken, om zo in het heden verder te kunnen. Dat verleden, documentatie genoemd, helpt iedereen, ook degene die het allemaal al weten. Want door die documentatie elke keer bij te houden, denk je toch even aan dat ene punt dat je anders zou zijn vergeten en dat in het verdere traject of in productie tot vertraging/onverantwoorde risico's zou leiden.

Goed ingewerkte teams zijn in staat zeer snel tot zeer goede resultaten te komen, maar wat als dat team een keer stopt? Vaak is het toch een soort van project dat maar een bepaalde tijd actief is. Je moet dat Agile team dus in leven houden en DevOps zal hier ook wat mee moeten. Een standaard manier van werken helpt en hier zijn templates weer een onderdeel van.

Herhalen

De echte kracht van de nieuwe wereld zit hem in de sprints/releases/iteraties/incrementen en het elke keer weer herhalen van een cyclus. De lengte van een cyclus is minder van belang. Dezelfde dingen elke keer weer doen (dus gebruikmakend van templates), maakt een team sneller en voorspelbaarder. Vooral dat voorspelbaarder verhoogt de kwaliteit.

Maar hoef ik dan geen testplan meer te maken? Jazeker wel; je moet nog steeds nadenken over kwaliteitsattributen, maar je zal merken dat dat ook binnen één minuut kan. Instant testmanagement wordt mogelijk.

Goede templates en voorbeelden doen wonderen (vooral in doorlooptijd), in welke vorm en hoe uitgebreid (waterval, Agile en allerlei tussenvormen) dan ook. Het gaat erom dat het binnen de context van de klant werkt. Heel vernieuwend dus… ←

TESTSTRATEGIE MET BEHULP VAN HEURISTIEKEN: EEN PRAKTISCHE AANPAK VOOR IEDEREEN!

Door Huib Schoots ● hsc@improveqs.nl @huibschoots

Een gedegen teststrategie is de basis voor ieder testtraject. Het geeft antwoord op de vragen: waarom testen we en hoe testen we? Een teststrategie helpt de tester de juiste dingen te doen, geeft testen structuur en geeft inzicht in de testdekking.

Problemen in testen

In mijn dagelijkse praktijk heb ik veel projecten gedaan, aangestuurd, geadviseerd en van dichtbij mogen bekijken. Uiteraard heeft vooral het testgedeelte mijn voornaamste aandacht. Wat me opvalt, is dat veel testtrajecten niet of nauwelijks met een teststrategie werken, ondanks dat er een hoofdstuk teststrategie is opgenomen in het testplan. Ik zie een aantal problemen in de praktijk met testen die gerelateerd zijn aan teststrategie:

- Testen wordt gezien als een aantal opeenvolgende fasen of activiteiten.
- Testers zien de teststrategie als een document of een fase in plaats van voortschrijdend inzicht.
- De vertaling van een PRA naar een concrete teststrategie is lastig.
- De meeste teststrategieën zijn vaag en gebruiken relatieve inspanning. Ze geven geen inzicht in wat testers moeten doen.
- Testers hebben geen idee wat ze nu wel en niet geraakt hebben. Ik heb het dan over het 'echte product' en niet over de specificatie of requirements. Hoe maak je dat inzichtelijk?
- Het toepassen van een testtechniek is een recept geworden. Altijd toepassen van classification tree, ook al laat de situatie dat niet toe: bijvoorbeeld gedrag van een product in een classification tree stoppen.
- Er is geen (duidelijk) verband tussen de informatiebehoefte, de testmissie, de teststrategie en de testen die uitgevoerd worden.

Teststrategieën verdwijnen helaas nog veel te vaak in een la. Dit artikel geeft handvatten om een heldere en bruikbare teststrategie te maken, die wel gebruikt zal worden! Maar laten we eens een paar stappen terug doen. Wat is testen eigenlijk? En waarom testen we?

> Sound strategy starts with having the right goal
>
> (Michael Porter)

Wat is testen? En wat is de missie van testen?

Testen verzamelt informatie en geeft inzicht in de status van een product om beslissingen te informeren (Jerry Weinberg, zie ref 1). Een algemene doel van testen is om informatie te verzamelen over dingen die belangrijk zijn. Door met onze stakeholders te praten en hen vragen te stellen, komen we er achter wat zij willen weten en wat zij belangrijk vinden. Testen kan daardoor veel verschillende missies [ref 2] hebben:

- belangrijke bugs vinden;
- de kwaliteit van het product beoordelen;
- managers inzicht geven in de voortgang van het project;
- managers informatie geven voor vrijgavebeslissingen;
- informatie geven voor het voorspellen en beheersen van de kosten van productondersteuning;
- controle van de implementatie van requirements (conformiteit specificaties); ➔

- helpen bij het verbeteren van de productkwaliteit;
- … enzovoorts.

Testen is dus informatie verzamelen voor verschillende missies. De meeste testers zijn niet gewend om op dit niveau over testmissie na te denken en hanteren gewoon steeds dezelfde impliciete testmissie. Daarom een voorbeeld van een wat minder gebruikelijke testmissie: 'het beoordelen van een product van een bedrijf dat misschien door jouw bedrijf overgenomen wordt.' Hierbij is het eenvoudig voor te stellen dat er anders getest moet worden.

Teststrategie

> There is nothing so useless as doing efficiently that which should not be done at all
>
> (Peter Drucker)

Daarna is het belangrijk om de juiste methode te bepalen om de gevraagde informatie te vinden. De testmissies hebben invloed op de teststrategie omdat verschillende testmissies een andere aanpak vragen (lees: een andere teststrategie). Het doel van de teststrategie is om het testen te sturen om de missies te bereiken. Het bestaat uit richtlijnen die beschrijven wat te testen en hoe te testen. De strategie informeert in eerste instantie de stakeholders en het testteam. Later gebruiken we de strategie, of beter gezegd bouwen we de strategie uit zodat we de testen kunnen uitvoeren.

Wat maakt een teststrategie tot een goede teststrategie? Een goede teststrategie is onder andere: [ref 3]

- Context-gerelateerd. Iedere situatie is anders. Ieder project is anders. Een teststrategie moet op maat gemaakt worden zodat het perfect past in de context.
- Specifiek. Niemand heeft iets aan vage statements. Hoe specifieker de teststrategie, hoe beter de stakeholders het begrijpen en hoe makkelijker het is om de testen te ontwerpen en ze uit te voeren.
- Risico-gebaseerd. Het belangrijkste risico dient als eerste getest te worden en verdient in veel gevallen de meeste testinspanning.
- Afwisselend & divers. Iedere methode of techniek is gericht op het vinden van andere informatie en vind ook andere bugs.
- Praktisch. Testen moeten uit te voeren zijn en bij voorkeur zo makkelijk mogelijk.
- Gedragen. Het is belangrijk dat de stakeholder erachter staan.
- Gerechtvaardigd. Iedere test die je uitvoert, moet bijdragen aan het doel en waarde toevoegen. Ook het kostenaspect speelt hiermee: is het de moeite waard?
- Flexibel. Een teststrategie verandert voortdurend door nieuwe inzichten, gevonden informatie en veranderende prioriteiten. Bovendien weten we aan het begin van het project nog lang niet alles en zal de nieuwe informatie ook invloed hebben op onze strategie.

Teststrategie met heuristics

> The essence of strategy is choosing what not to do
> (Michael Porter)

Oké, maar hoe maak je dan een goede teststrategie? Een teststrategie zou een leidraad moeten zijn voor de tester om testen uit te voeren die voldoen aan de missies. Maar het wordt ook gebruikt om de testinspanning te bepalen en de testplanning te maken. Om een goede teststrategie te maken moeten we aan ontzettend veel dingen denken. Ik gebruik het Heuristic Test Strategy Model van James Bach en Michael Bolton om me te helpen. Het doel van dit model is om me te helpen aan alle belangrijke aspecten te denken als ik test. ➔

Wat zijn heuristics?

Heuristieken zijn feilbare methoden voor het oplossen van een probleem of het nemen van een beslissing. Het zijn snelkoppelingen om complexe problemen op te lossen of kortweg vuistregels, ezelsbruggetjes of geheugensteuntjes. Ze worden gebruikt om binnen een redelijke tijd voor moeilijke problemen haalbare oplossingen te bepalen die goed genoeg zijn. Op de Engelstalige Wikipedia vond ik deze definitie [ref 4]: 'Heuristiek zijn ervaring-gebaseerde technieken voor het oplossen van problemen, om een oplossing te leren en ontdekken die niet gegarandeerd optimaal is. Waar een uitputtende zoektocht onpraktisch is, worden heuristische methoden gebruikt om het proces te versnellen bij het vinden van een bevredigende oplossing via mentale snelkoppelingen die de cognitieve belasting bij het nemen van een beslissing vergemakkelijken.'

Wil je meer weten over heuristics? Lees dan deze blog posts van Michael Bolton [ref 5] en James Bach [ref 6]:

- 5 - http://www.developsense.com/blog/2012/04/heuristics-for-understanding-heuristics/
- 6 - http://www.satisfice.com/blog/archives/462

Heuristic Test Strategy Model

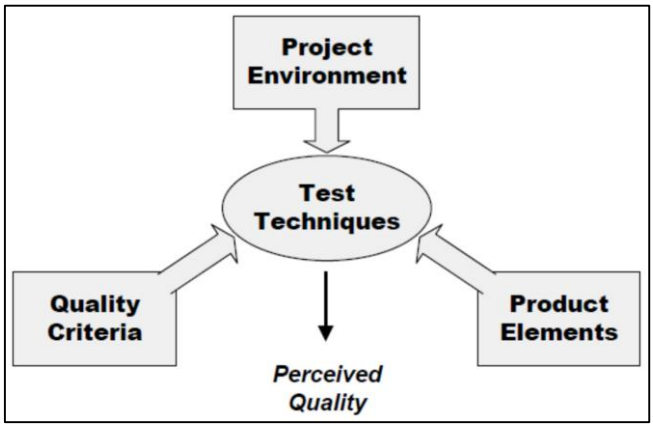
Het heuristic test strategy model (HTSM, [ref 7]) is ontwikkeld door James Bach en bestaat uit vier lijsten met heuristieken die helpen bij het opstellen van een gedegen teststrategie.

Projectfactoren: context-factoren die van cruciaal belang zijn om te kunnen beslissen welke specifieke testen uitgevoerd moeten worden.

- Missie: Je doel in dit project, zoals afgesproken met je klanten.
- Informatie: informatie over het product of project die nodig is voor het testen.
- (Relatie met) Developers: hoe werk je samen met de programmeurs en de andere teamleden?
- Testteam: wie zal de testen uitvoeren of ondersteunen?
- Apparatuur en tooling: hardware, software of documenten die nodig zijn om het testen uit te voeren.
- Planning: de volgorde, duur en synchronisatie van project gebeurtenissen.
- Testitems: het te testen product.
- Deliverables: de waarneembare producten van het testproject.

Productelementen: een product bestaat uit meerdere elementen of producteigenschappen. De verschillende gezichtspunten moeten helpen de juiste testvragen te formuleren.

- Structuur: verschillende (onder)delen van het fysieke product.
- Functionaliteit: de functies, alles wat het product doet.
- Data: alles wat wordt verwerkt.
- Interfaces: elk kanaal waarmee het product interacteert.
- Platform: alles waar het product van afhankelijk is.
- Operaties: hoe het product gebruikt wordt.
- Tijd: elke relatie van het product met tijd.



Kwaliteitsattributen: bepaalde eisen die aangeven wat of hoe het product zou moeten zijn. Elk van de items op deze lijst kan worden gezien als een potentieel risicogebied. ➔

- Geschiktheid: kan het de vereiste functies uitvoeren?
- Betrouwbaarheid: werkt het goed en is het bestand tegen fouten?
- Bruikbaarheid: hoe gemakkelijk is het te gebruiken?
- Charisma: hoe aantrekkelijk is het product?
- Veiligheid: hoe goed is het product beveiligd?
- Schaalbaarheid: hoe goed kan het product uitgebreid of ingekrompen worden?
- Compatibiliteit: hoe goed werkt het met externe componenten en configuraties?
- Prestaties: hoe snel is het?
- Installeerbaarheid: hoe gemakkelijk kan worden geïnstalleerd?
- Ontwikkeling: hoe goed kunnen we het product maken, aanpassen en testen?

Software Quality Characteristics [ref 8] is een uitgebreidere lijst. Deze is ook in het Nederlands vertaald [ref 9] en is online te vinden. De ISO25010 (voorheen ISO9126) lijst met kwaliteitskenmerken kan ook als alternatief gebruikt worden.

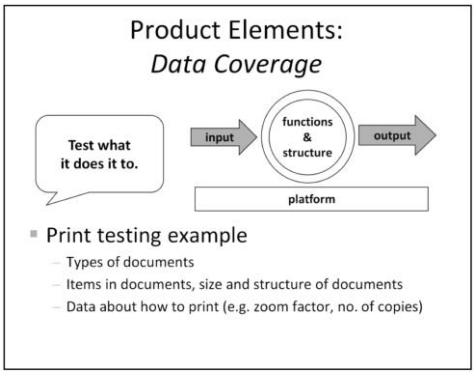
Algemene testtechnieken: een 'algemene techniek' is een techniek die eenvoudig en universeel genoeg is om te gebruiken in zeer verschillende omstandigheden.

- Functietesten: test wat het kan doen.
- Domeintesten: deel de gegevens op in logische verzamelingen.
- Stresstesten: overweldig het product.
- Flowtesten: doe de ene handeling na de andere.
- Scenariotesten: test een meeslepend scenario.
- Claimtesten: controleer elke bewering over het product.
- Gebruikerstesten: betrek de gebruikers.
- Risicotesten: stel je een probleem voor en ga ernaar op zoek.
- Automatische controle: check een miljoen verschillende feiten.

Vele specifieke technieken (zoals de technieken in TMap Next) zijn gebaseerd op één of meer van deze negen. Zo is de Data Combinatie Test een vorm van domeintesten, de Elementaire Vergelijkingen Test een vorm van functietesten en Error Guessing een risicotest.

Bekijk deze korte video waarin Michael Bolton het gebruik van het HTSM uitlegt.

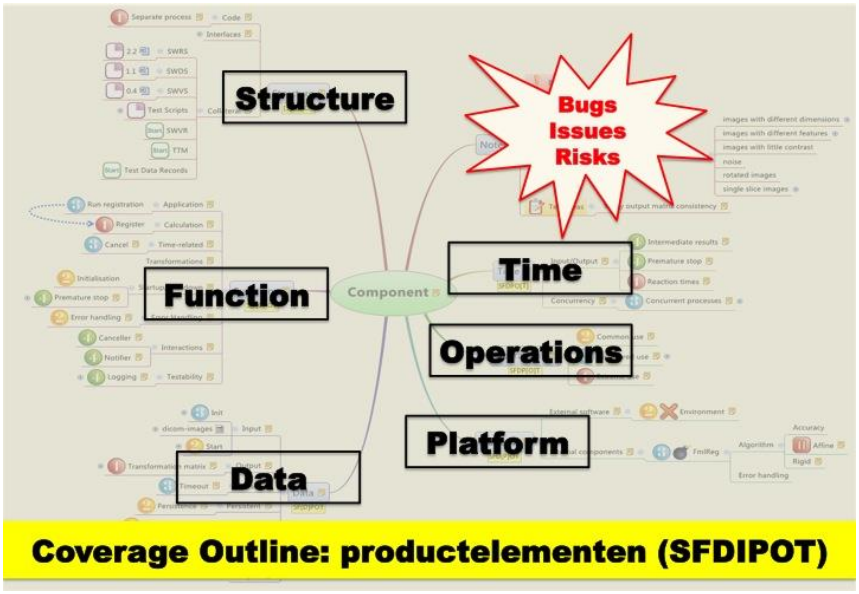
- https://www.youtube.com/watch?v=SNHWCD2qg3U



Hoe gebruik je het HTSM?

De projectfactoren helpen je bij een analyse van de context. Het Heuristic Test Strategy Model bevat vervolgens zeven verschillende perspectieven op een product zoals structuur, functionaliteit, data (figuur rechts), interfaces, platform, operaties en tijd. Het gaat hier om een coverage model dat uitgebreid is beschreven in Rapid Software Testing [ref 10]. ➔

Er zijn vele manieren waarop je de informatie over productelementen kan modelleren. Onderstaand is een voorbeeld van een mind map van Ruud Cox. Andere vormen zijn ook mogelijk, het gaat er uiteindelijk om dat het onderliggende product goed gemodelleerd wordt.



Naast productelementen maken de kwaliteitsattributen inzichtelijk welke aspecten van het product getest moeten worden.

Testtechnieken

Met dit gevulde model in de hand kan een tester dan bepalen welke testtechnieken het meest geschikt zijn. Om de dekking en diepgang inzichtelijk te maken kan hiermee ook aangegeven worden welke delen van het product en welke kwaliteitsattributen (of aspecten van het product) daarmee geraakt worden en welke niet. De keuze voor een bepaalde testtechniek wordt ingegeven door de projectfactoren te beschouwen en vervolgens per productelement, per kwaliteitsattribuut te bedenken welke testtechniek toegepast kan worden.

Diversifieer

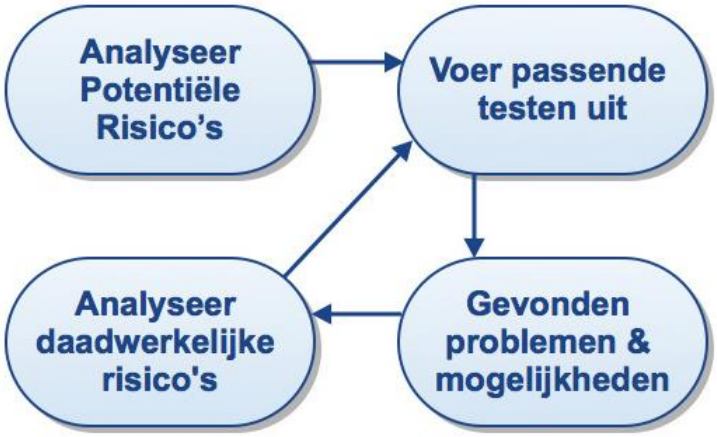
Er is niet één techniek die alle bugs vindt. Verschillende technieken vinden verschillende bugs. Bovendien is het zo dat geen enkele techniek perfect toegepast kan worden. Gebruik daarom veel verschillende invalshoeken, benaderingen en technieken. Zelfs al wordt geen enkele volledig uitgevoerd. Het is beter om tien technieken half toe te passen, dan één helemaal perfect [ref 11]. Datzelfde geldt natuurlijk ook voor testideeën: het is in de meeste gevallen beter om veel testideeën te hebben die veel verschillende testen kort en snel doen, dan één zeer uitgebreid en bijna perfect uitgewerkt idee.

Dekking en diepgang

Hoe hoger het risico, hoe belangrijker de uit te voeren test is. Dat hoeft niet per se te betekenen dat de testen ook meer diepgang hoeven te hebben of dat er een formele(re) testtechniek wordt ingezet. Onderstaande tabel is een voorbeeld hoe een bepaalde testspecificatietechniek direct gekoppeld wordt aan de hoogte van het risico. ➔

Laag risico	Gemiddeld risico	Hoog risico
Test use cases	Equivalentieklassen	Beslissingstabellen
Error guessing	Test use cases	Equivalentieklassen
Statement coverage 60%	Statement coverage 70%	Decision coverage 70%

Ik denk dat dit in veel gevallen een onjuiste benadering is! Ik kan me vele situaties voorstellen waarin een hoger risico met minder testen en een informelere testtechniek toch meer zekerheid geeft. Denk bijvoorbeeld aan een login van een intranet applicatie. Die is enorm belangrijk omdat, als dat niet werkt, de gehele applicatie niet bereikbaar is. De login is vaak relatief eenvoudig te testen. Het testen van de lay-out en syntax van alle schermen erachter is veel meer werk.



De juiste test is van veel factoren afhankelijk en kan op vele manieren worden gediversifieerd. Een manier om hiermee te variëren is de gekozen testtechniek. Andere manieren zijn: het aantal testtechnieken die gebruikt worden op hetzelfde deel, de hoeveelheid combinaties die getest worden, de diepgang van de testen, een vastgestelde timebox, et cetera.

Van klein naar groot

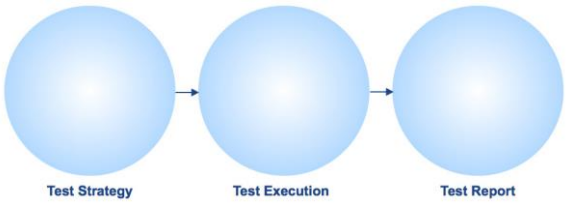
Een teststrategie groeit gedurende het project: we komen steeds meer te weten over het product en we krijgen een steeds beter beeld over de risico's. Hierdoor zullen prioriteiten veranderen, te testen onderdelen aangevuld worden, et cetera. Tijdens het project leren we en daardoor groeit de strategie, van globaal naar steeds meer detail.

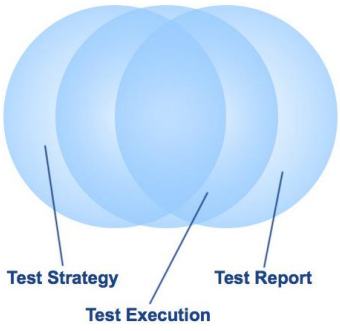
Een testtraject begint met een risicoanalyse. Voor de gedefinieerde risico's worden testen uitgevoerd waardoor we weer leren en meer te weten komen over de daadwerkelijke situatie. We leren niet alleen over problemen, maar ook over mogelijkheden die weer tot nieuwe inzichten kan leiden.

Teststrategie versus testuitvoering versus rapportage

De teststrategie is een belangrijk onderdeel van testen en kan niet los gezien worden van de testuitvoering en de rapportage. Een goede teststrategie geeft de testuitvoering richting en op alle onderdelen van de teststrategie zal later gerapporteerd worden. ➔

Veel testers zien de strategie als een los onderdeel zoals de afbeelding rechts. Ik zie testen als een integraal geheel en kan de test strategie dan ook niet los zien van de testuitvoering en de rapportage (afbeelding links). Voor mij is de teststrategie een concrete en specifieke uitwerking van wat ik ga testen. Tijdens de testuitvoering voeg ik daar meer details aan toe en wordt de strategie uitgebreid. Uiteindelijk zal mijn testrapportage ook rechtstreeks gebaseerd zijn op de onderdelen uit de teststrategie.





Testideeën

Het is vaak lastig om goede testen te verzinnen. Daarom gebruik ik tijdens het opstellen van de teststrategie, het ontwerpen van testen en de testuitvoering vele checklists die me helpen om snel ideeën te genereren. Deze checklists zijn eigenlijk ook heuristieken.

Enkele voorbeelden zijn:

- The Little Black Book on Test Design van Rikard Edgren met daarin 37 sources for Test Ideas maar het hele (gratis) ebook bevat vele bronnen ter inspiratie voor testers. - http://www.thetesteye.com/papers/TheLittleBlackBookOnTestDesign.pdf
- Test Heuristics Cheat Sheet van Elisabeth Hendrickson - http://testobsessed.com/wp-content/uploads/2011/04/testheuristicscheatsheetv1.pdf
- You Are Not Done Yet checklist van Michael Hunter - http://www.thebraidytester.com/downloads/YouAreNotDoneYet.pdf
- 36 days of web testing van Rob Lambert als bron voor testideeën voor websites - http://thesocialtester.co.uk/wp-content/uploads/2012/10/36DaysOfWebTesting.pdf
- Op mijn blog zijn meer bronnen te vinden die als inspiratie kunnen dienen. - http://www.huibschoots.nl/wordpress/?page_id=441#ideas

Ten slotte

Een goede teststrategie neemt ook unit testen (of hoe je de testen ook wil noemen die developers uitvoeren) mee. Testen is een teamsport en unit testen zijn de basis van het testen: zorg dus dat je weet wat de ontwikkelaars doen! En help ze daarbij.

Visualiseer je teststrategie! Gebruik bijvoorbeeld mind maps om een teststrategie vast te leggen. Maar er zijn veel grafische manieren om je testen te ondersteunen. Maak daar gebruik van, want testen is vaak complex en door te tekenen, diagrammen te maken of op andere manieren je (denk)werk te visualiseren, zal het meestal makkelijker gaan.

Lees eens wat meer over 'test framing [ref 12]'. Test framing is een belangrijke vaardigheid die ons helpt om op een logische, samenhangende en toch snelle manier onze testen te rechtvaardigen: bij elke test kunnen aangeven wat het verband is met de testmissie(s). Het doel van de test framing is om te kunnen om heldere en geloofwaardige antwoorden te geven op vragen als 'Waarom voer je deze test uit?' of 'Hoe voer je deze test uit?'.

Houd rekening in je testen met testbaarheid [ref 13] en vraag hierom vanaf het eerste moment! Testbaarheid maakt het testen sneller en praktischer. Bijvoorbeeld door de juiste ontwerptechniek of diagrammen te vragen aan ➔

ontwerpers, kunnen testontwerpen efficiënter gemaakt worden. Door bij programmeurs om logging te vragen, kunnen testresultaten sneller geanalyseerd worden. Ook ondersteunen tools zoals 'perlclip [ref 14]' je testen. Neem testbaarheid mee in je teststrategie en praat erover met je team!

Probleemoplossing

Er is geen standaard recept voor het maken van een strategie of het uitvoeren van testen. Heuristieken helpen bij het oplossen van problemen. Zie de teststrategie als een oplossing voor een complex probleem: 'Hoe voldoen we aan de informatiebehoefte van de stakeholders op een zo efficiënt mogelijke manier?'. Dit probleem is in iedere situatie anders en vraagt daarom iedere keer om een andere oplossing. Ik hoop dat testers na het lezen van dit artikel inzien dat 'probleemoplossing [ref 15]' beter bij testen past dan 'recept denken'.

Met dank aan Ruud Cox voor de vele discussies over dit onderwerp en zijn review, Joep Schuurkes voor een uitgebreide review, Rikard Edgren voor inspiratie en uiteraard James Bach en Michael Bolton voor het delen van hun kennis over HTSM.

Referenties

1 - http://www.dorsethouse.com/books/perf.html

2 - http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf (slide 69 e.v.)

3 - http://thetesteye.com/blog/2013/09/what-is-a-good-test-strategy

4 - http://en.wikipedia.org/wiki/Heuristic

7 - http://www.satisfice.com/tools/htsm.pdf

8 - http://thetesteye.com/posters/TheTestEye_SoftwareQualityCharacteristics.pdf

9 - http://dewt.files.wordpress.com/2013/03/thetesteye_softwarekwaliteitkenmerken1.pdf

10 - http://www.satisfice.com/info_rst.html

11 - http://www.kaner.com/pdfs/test_docs_pnsqc.pdf

12 - http://www.developsense.com/resources/TestFraming.pdf

13 - http://www.satisfice.com/tools/testable.pdf en http://www.developsense.com/blog/2009/07/testability/ en

14 - http://www.satisfice.com/tools.shtml

15 - http://nl.wikipedia.org/wiki/Probleemoplossing ←

DE AAP UIT DE MOUW

Door Daisy de Joode ● d.m.joode@avisi.nl

Hoe stel jij jezelf op als je een bug meldt bij een engineer? Vind jij het gedrag van jouw collega soms vol zitten met bugs? Onze menselijke reacties zijn beestachtig beïnvloedbaar. Lees meer over een mainframe met een hondenleven en een DDOS-aanval gestart door een viervoeter.

Software test engineering. Een prachtig vak waarbij je alle ins en outs van een bepaald stuk software, database of ander digitaal fenomeen mag ontdekken en beoordelen op functionaliteit en kwaliteit. Het analyseren van systeemgedrag en de vergelijking trekken met het gewenste gedrag of hier suggesties op aandragen is wat wij graag doen. Maar hoe zit het eigenlijk met het menselijk gedrag? Hoe stel jij jezelf op als je met een lijst bevindingen (lees: slecht nieuws) naar een engineer toe stapt? Vind jij het gedrag van jouw collega soms ook vol zitten met bugs? En zou het niet fijn zijn om een lijst met foutmeldingen te hebben waarin de oorzaak staat benoemd van zo'n bug in plaats van dat je zelf moet raden waarom jouw collega zo reageert?

Geloof me, menselijk gedrag valt net zo te beïnvloeden als systeemgedrag. En zo'n lijst met foutmeldingen bestaat. Ontdek hoe jij sterker in je schoenen kunt staan in situaties waarin je het tegen een 'developer zwaargewicht' moet opnemen. Om hier begrip en inzicht in te krijgen gaan we terug naar onze 'roots' en die vinden we terug in het dierenrijk. Aan de hand van vier gedragstyperingen neem ik je mee op een ontdekkingsreis over jezelf en anderen.

NAS versus Mainframe

Een mooi voorbeeld is het gedrag van honden. Wanneer het gaat om leiderschap, is het niet enkel uiterlijke verschijning dat daar een belangrijke rol in speelt. Nee, zij hebben een geraffineerd gedrag ontwikkeld dat zelfs zo ver gaat dat hondjes niet groter dan een NAS-server de baas kunnen zijn over een soortgenoot 'maatje mainframe'. Om tot dit resultaat te komen dien je een inschatting te maken van degene die je tegenover je hebt. Het is zaak om de juiste moves te maken die de reactie van jouw tegenspeler – voor jou – positief beïnvloedt. Als een Chihuahua deze leiderschapsmoves kan maken, dan kan jij dat ook.

DDOS- aanval

Naast het vertonen van gedrag dat leidt tot leiderschap, zijn er ook situaties waarin juist de aanval wordt ingezet. Het 'op je achterste poten gaan staan' wordt in het dierenrijk veelvuldig gebruikt. Denk aan steigerende dominante paarden, twee herten in conflict, maar ook aan een prairiehondje dat middels fysiek machtsvertoon zijn zin doordrijft. Ook jij, ik of een collega maken zo nu en dan gebruik van deze techniek. Op zo'n moment zorg je voor een distributed denial of service doordat je de ander bewerkt en beperkt vanuit meerdere perspectieven: een zelfverzekerde lichaamshouding, stemverheffing en een scherpe mening. ➔

Firewalling

De derde typering is defensief gedrag en brengt dito mechanismen met zich mee. Neem bijvoorbeeld het stekelvarken. Bij gevoel van bedreiging zet hij zijn stekels omhoog in de hoop dat de aanvaller afdruipt. Deze 'firewall' werkt bij mensen precies hetzelfde. Stel, je spreekt iemand aan omdat het opgeleverde werk niet aan de minimale kwaliteitseisen voldoet. Grote kans dat de aangesproken persoon in de verdediging schiet. Muren gaan omhoog en zintuigen staan op scherp, waardoor jij je overrompeld voelt en wellicht afdruipt of je mening herziet. Dit is natuurlijk niet wat je wilt.

RAID 1 – Mirroring

Tot slot de laatste gedragsvorm die we kunnen tegen komen in een dialoog: aanpassend - mirroring - gedrag. Je mening spiegelen aan de omgeving is een veilige, passieve wijze van jezelf uiten. Zolang je niet in het middelpunt van de aandacht staat, voel je jezelf veilig.

Een kameleon stelt zijn hachje veilig door zijn kleur op de omgeving aan te passen en ontpoppen nachtvlinders zich als bijna volwaardige RAID 1-replica's van bladeren. De confrontatie aangaan met je collega die zich voortdurend aanpast aan de mening van een ander is makkelijk te overwinnen, maar dat wil je niet. Je bent niet op zoek naar een ja-knikker, maar je zoekt de dialoog en vraagt om iemands persoonlijke mening.

Interactie

Dit waren in hoofdlijn de vier gedragstyperingen die jij op je werk kunt tegenkomen tijdens een dialoog. Gedrag is altijd het gevolg van interactie tussen een aangeboren - uit het dierenrijk afkomstig - mechanisme en de aanwezige omgevingsinvloeden.

Zo. De aap is nu uit de mouw. Ben je nieuwsgierig geworden? Wil je weten over welk dierlijk gedrag jij en je collega's beschikken en hoe je dit kunt beïnvloeden? Meld je dan aan voor de workshop 'Mijn collega is een aap'.

Bron

Roos van Leary Nieuwenhuis, M.A., The Art of Management (the-art.nl), ISBN-13: 978-90-806665-1-1, 2003-2010.

←

TESTEN IN ONDERWIJSLAND MET SERVICE VIRTUALISATIE

Door John van der Molen (links) ● john.van.der.molen@testwerk.nl

Rix Groenboom (midden) ● rix.groenboom@parasoft.nl

Bas Dijkstra (rechts) ● bas.dijkstra@oelan.nl

Studielink is de centrale schakel in de gegevensuitwisseling tussen hoger onderwijsinstellingen en leveranciers van studentinformatiesystemen (SIS) aan de ene kant en de desbetreffende uitvoeringsinstantie van de overheid aan de andere kant. Het testteam van Studielink is afhankelijk van de beschikbaarheid van representatieve testomgevingen aan de kant van de overheid voor het succesvol kunnen testen van deze gegevensuitwisseling. Er zijn voor dit doel echter beperkt testomgevingen beschikbaar voor Studielink, waardoor niet alle testomgevingen van Studielink en daarmee niet alle onderwijsinstellingen en SIS-leveranciers in verbinding staan met de gewenste testomgevingen binnen de overheid. Hoe heeft service virtualisatie Studielink in staat gesteld om succesvol gesimuleerde testomgevingen beschikbaar te stellen voor de uitvoer van ketentesten in de informatieketen in het hoger onderwijs?

De opdracht

De applicatie Studielink, welke wordt ontwikkeld en beheerd door de gelijknamige stichting, vormt een belangrijke schakel tussen de instellingen voor hoger onderwijs in Nederland en de desbetreffende uitvoeringsinstantie van de overheid. Via het web-portaal van Studielink (www.studielink.nl) regelen alle studenten digitaal hun inschrijving in het hoger onderwijs. Studielink verzamelt alle benodigde gegevens en geeft deze informatie door aan de onderwijsinstellingen voor het inschrijfproces bij de instelling.

Studielink koppelt onder meer de studentinformatiesystemen (SIS, meer dan tien soorten) van de onderwijsinstellingen (meer dan vijftig) aan de juiste overheidswebservices. Via deze koppeling worden diverse gegevens van studenten uitgewisseld tussen de onderwijsinstellingen en de overheid, waaronder:

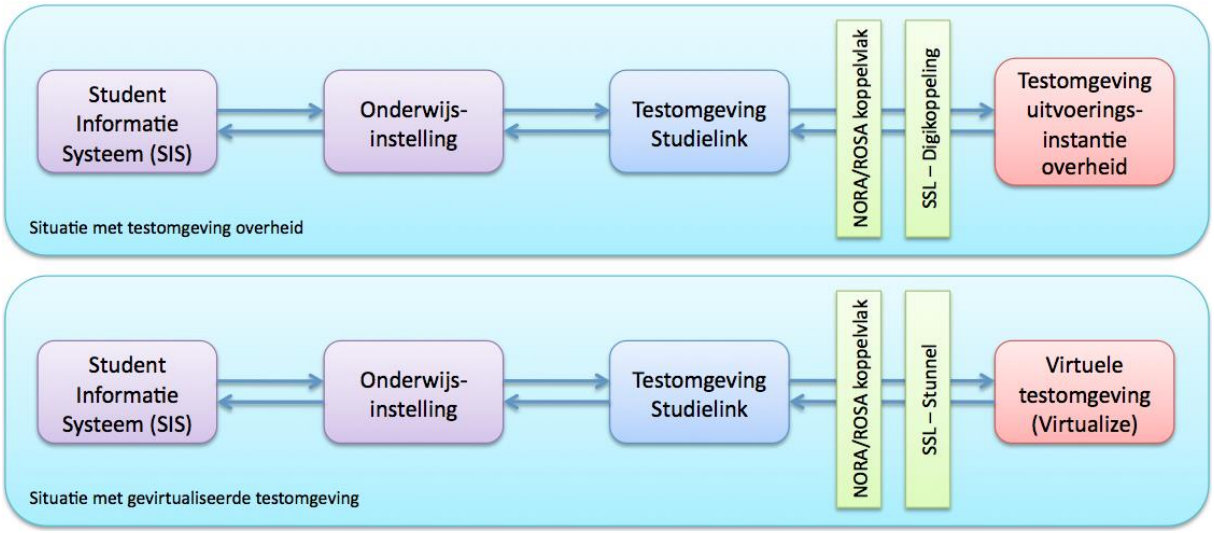
- Persoonsgegevens (zowel GBA- als niet-GBA-gegevens);
- Gegevens over de genoten vooropleidingen;
- Informatie over de hoogte van het collegegeld;
- Indicatie van de bekostiging van een opleiding voor de onderwijsinstelling (deze is bepalend voor de jaarlijkse toelage die een onderwijsinstelling ontvangt);
- Inschrijvingen voor en resultaten naar aanleiding van opleidingen.

Studielink heeft de beschikking over een aantal testomgevingen met een koppeling met onderwijsinstellingen. Aan de kant van de overheid zijn slechts beperkt testomgevingen beschikbaar, met als gevolg dat er slechts één ➔

testomgeving van Studielink in verbinding staat met een daadwerkelijke overheidstestomgeving. Een koppeling tussen onderwijsinstellingen en SIS-leveranciers aan de ene kant en een testomgeving van de overheid aan de andere kant is echter van groot belang voor het goed en volledig kunnen uitvoeren van de gewenste ketentestscenario's. Deze scenario's bestaan uit een continue XML-berichtuitwisseling tussen onderwijsinstellingen en de overheid via Studielink. Als een van de ketenpartners in deze keten niet beschikbaar is, kunnen testscenario's niet of niet volledig worden uitgevoerd.

De oplossing

Om toch een volledige keten beschikbaar te stellen voor de SIS-leveranciers en onderwijsinstellingen op de testomgevingen van Studielink moest er naar een andere oplossing gezocht worden. Deze diende zich aan met de introductie van service virtualisatie, waarmee het gedrag van de testomgeving van de overheid kan worden gesimuleerd. Hierbij wordt gebruik gemaakt van de service virtualisatie-oplossing Virtualize. Met deze tool is Studielink erin geslaagd om, op basis van de WSDL-definitie van het koppelvlak tussen Studielink en de overheid, binnen afzienbare tijd een ketentestomgeving met een gesimuleerde koppeling met de overheid beschikbaar te stellen.



De uitvoering

In de eerste fase van het project zijn de infrastructurele hobbels uit de weg geruimd en zijn de eerste berichtstromen gevirtualiseerd. Communicatie tussen Studielink en de virtuele overheids-testomgeving verloopt via een met SSL-certificaten beveiligde verbinding. Deze worden in de virtuele testomgeving afgehandeld door Stunnel. Vervolgens worden de berichten die worden verzonden vanuit Studielink inhoudelijk verwerkt door Virtualize, waarna de antwoordberichten wederom via Stunnel beveiligd naar Studielink worden teruggestuurd.

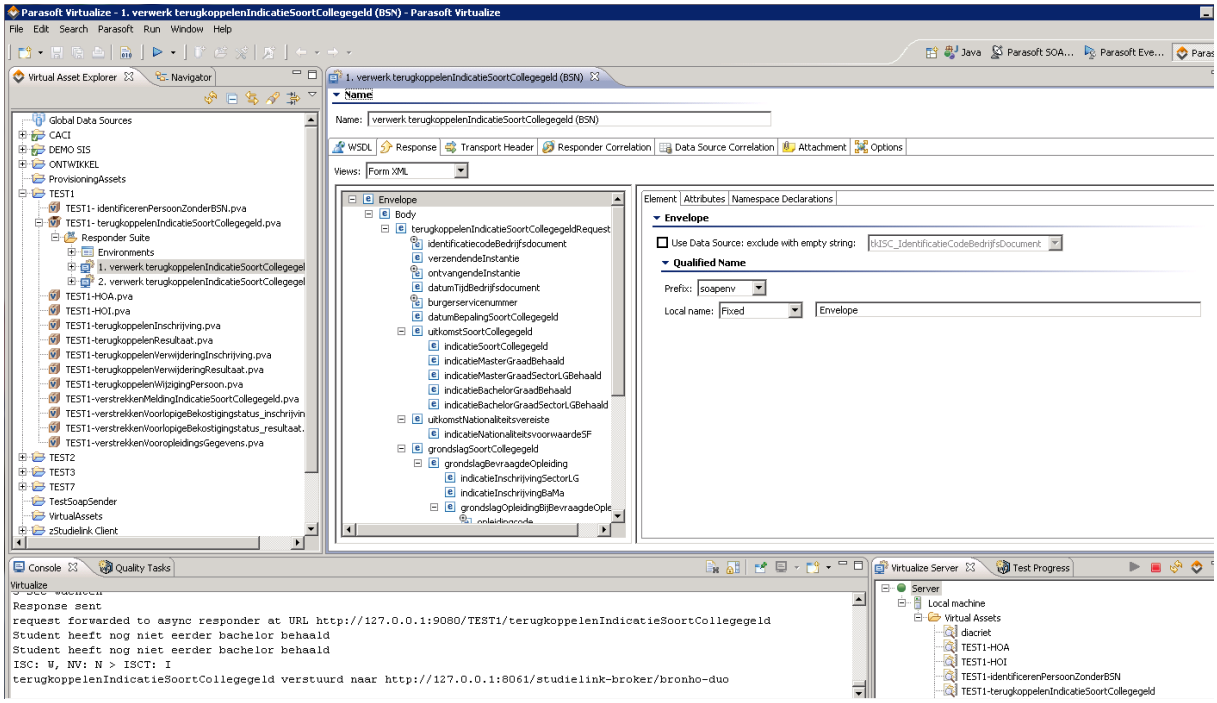
Het gebruik van gevirtualiseerde testomgevingen is Studielink zo goed bevallen dat zij in het kader van de meest recente release van het berichtenverkeer van en naar de overheid graag de functionaliteit van deze testomgevingen wilde uitbreiden. Hiertoe werd het aantal ondersteunde berichtstromen fors uitgebreid en werd de virtuele testomgeving voorzien van de mogelijkheid om data persistent op te slaan om later in het proces te hergebruiken. Dit resulteerde in een virtuele testomgeving die de functionaliteit van de daadwerkelijke testomgeving zeer dicht benaderde. ➔

De resultaten

Door de inzet van service virtualisatie is Studielink, ondanks het beperkte budget, in staat geweest om op alle testomgevingen gebruik te maken van een overheidstestomgeving met een realistische simulatie van de gewenste berichtenstromen. Ook konden de technische aspecten van de koppeling, zoals het gebruik van certificaten, succesvol worden gesimuleerd. Dit heeft in hoge mate bijgedragen aan het succesvol kunnen uitvoeren van de voor de Studielink-release geplande testscenario's.

Of, zoals verwoord door het testteam van Studielink:

'Met de virtuele testomgeving kunnen we, ongeacht de (on)beschikbaarheid van de testomgevingen bij de overheid, onderwijsinstellingen en SIS-leveranciers de zekerheid bieden om over de volledige Studielink-keten, testgevallen te kunnen uitvoeren.' ←



HELP, TESTERS WORDEN STEEDS TECHNISCHER!

Door Jacob Hooghiemstra ● J.Hooghiemstra@cimsolutions.nl

De laatste jaren worden we overspoeld met software die een technisch karakter heeft. SoapUI om webservices te testen en Toad om databases te raadplegen. Daarnaast zijn er SCRUM en DevOps als methoden op de markt gekomen die hun oorsprong in de techniek hebben. DevOps maakt af wat Agile begon: technologische en procesmatige integratie tussen ontwikkeling en beheer.
Wat vraagt dat van een test professional? Hoe houdt de tester grip op niet-technische zaken als usability testen, functioneel testen en gebruikersacceptatietesten? Wie houdt zich bezig met kwaliteitszorg en kwaliteitsmanagement of procesverbetering in het algemeen? Komen er robots die softwaretesten op eventuele defects? Of bouwen testers software die in staat is zichzelf te testen?

Verandering in testvak

Vroeger waren testers starre mensen die de hele dag testten, zonder rekening te houden met de rest van het project. Testen, testen, testen. Iets gevonden? Gooi het over de muur en laten 'ze' het maar oplossen. Discussie is niet mogelijk. De wetenschap heeft niet voor niets aangetoond dat autisten hele goede testers zijn. Tegenwoordig zijn testers veel pragmatischer. Communicatief vaardig en betrokkenheid, zijn competenties die van het grootste belang geworden zijn en exacte documentatie is dat een stuk minder.

De rol van ontwikkelaar is in de loop der jaren steeds groter geworden. De programmeurs zijn steeds mondiger geworden en binnen het projectteam zijn ze meer aanwezig. Binnen DevOps bestaat een rol uit een combinatie van meer professies: een ontwikkelaar met test skills of een functioneel beheerder met programmeerervaring. Applicaties bestaan steeds vaker uit losse componenten met (weer) verbindingen naar andere systemen. Hierdoor is de nadruk meer gaan liggen op het technische aspect van de software. De vraag om het technisch vraagstuk van losse componenten technisch goed te testen is daarmee ook groter. De hoeveelheid tijd die een tester krijgt, is er in de laatste jaren misschien wel beter op geworden, doordat testen nu meer vanzelfsprekend is, maar hoe wordt deze ingevuld? Testers worden meer gevraagd om meer technische onderdelen te testen en ontwikkelaars gaan meer mee-testen. Dit mee-testen komt met nog meer uitdagingen. Een bijeffect van Agile. ➔

Alleskunner

De laatste jaren is de functie Tester steeds meer een containerbegrip voor een diversiteit aan testen die worden uitgevoerd. In een project wordt de tester gezien als de persoon die zeurt en tijd van het project opbrandt. Vroeger was de tester de sluitpost van het project. Er ontstond een bewijsdrang om het bestaansrecht van de tester te bewijzen. Testers wisten overal wat van, want testen en kwaliteit kun je namelijk overal op loslaten. Langzaam werd de tester de spin in het web en was hij (of zij) degene met het totaalbeeld, de helicopterview.

Tegenwoordig is de tester, volgens het Paretoprincipe (wat in de volksmond ook wel de 80-20-regel genoemd wordt), 80% bezig met het testen van software en 20% met kwaliteit in algemene zin. De bewijsdrang is voor testers zo groot dat ze alle technische ontwikkelingen graag bij willen en moeten houden: kennis van SoapUI, inclusief de gehele technische infrastructuur, kennis van databases als SQL en Oracle met de relaties en unieke sleutels. Maar al deze kennis opdoen vraagt veel tijd en bijhouden is bijna onmogelijk. De intensieve studie hiervoor gaat ten koste van andere zaken. Het past domweg niet meer binnen de functie Tester oftewel vaststellen in hoeverre de software aan de eisen voldoet.

Back to Basic

Wordt het niet tijd om de techniek los te laten? Tijd om als testers weer te richten op andere belangrijke zaken. Testers zijn ontwikkelaars noch ontwerpers. Natuurlijk is het handig als de tester verstand van zaken heeft en weet waarover gesproken wordt. Maar een projectleider is ook geen ontwikkelaar. Sterker nog, testers pretenderen dat testen echt een vak apart is en ze daarbij geen technische kennis nodig hebben.

Natuurlijk moet het mogelijk zijn dat een tester zich wil specialiseren in berichtenverkeer, databasetesten of back-officesystemen; een technisch tester met de daarbij behorende soft skills en materie skills. Het maakt de technische tester niks uit of de lay-out wel goed is, als het maar functioneert. Ook moet het mogelijk zijn omdat de tester zich specialiseert in de zachte kant van testen, het begeleiden van gebruikersacceptatietesten. Daar horen ook andere soft skills en materie skills bij.

Kwaliteitsbewaker

Een van de activiteiten van een tester is om de kwaliteit van het gehele systeem of product te controleren. Een andere werkzaamheid is inzicht geven in, adviseren over de kwaliteit van de software en de daaraan gerelateerde risico's. Juist deze activiteiten zijn belangrijk en worden vaak niet door andere professies opgepakt. Prince2 onderkent risico als een van de minimale managementaspecten, TMAP Next gebruikt de Product Risico Analyse. Eigenlijk zou de nieuwe tester zich veel meer, zeg 80%, moeten richten op kwaliteit en 20% moeten functioneren in softwaretesten. Dit kan ook omdat steeds meer software geautomatiseerd of automatisch getest kan worden.

Een belangrijk punt waar veel IT-projecten tegenaan lopen, is de traditionele kloof tussen business en IT. Doordat eindgebruikers en ontwikkelaars in verschillende werelden leven, verloopt de communicatie over en weer dikwijls stroef. Met alle gevolgen van dien. Ze begrijpen elkaar niet goed en misvattingen liggen voortdurend op de loer. Daar ligt een gat en een kans voor de kwaliteitsmanager.

De kwaliteitsbewaker kan daarbij gebruikmaken van de standaarden en methoden die er al zijn, bijvoorbeeld uit de industrie zoals Six Sigma, IEEE 730 en ISO 9001.

NEN is gestart met het ontwikkelen van de Nederlandse praktijkrichtlijn (NPR) 5325 ‘Opleveren en overdragen van software’. Deze praktijkrichtlijn gaat over het beoordelen van kwaliteit van software en is met name gericht op onderhoud (maintenance) en herbruikbaarheid (portabiliteit) van software. ➔

Het bedrijfsleven wil in de toekomst steeds meer grip krijgen en houden op IT-projecten. Het percentage mislukte IT-projecten is nog steeds erg hoog. Het gat tussen IT en business is daar een belangrijke oorzaak van. De software die bedacht en ontwikkeld moet worden, moet een afspiegeling zijn van het bedrijfsproces. Deze vertaling brengt risico's met zich mee.

Bruggenbouwer

Testers zijn van oorsprong kwaliteitsmensen. Mensen die de wereld van business en de wereld van IT begrijpen en zo een toegevoegde waarde bieden aan het IT project. Het is goed om wat van de techniek af te weten, maar houdt daarbij wel focus op waar het primair om gaat. Als bruggenbouwer tussen twee werelden, met voldoende kennis om beide te begrijpen. Fit for purpose.

De kwaliteitsstandaarden en testmethoden geven handen en voeten om risico's te zien, te ontdekken en op waarde en impact te schatten. Dit geeft een beeld te weten wat te doen staat.

Voor testers de uitgesproken kans om daar wat aan de doen! ←

MAG HET EVEN WAT MINDER PROFESSIONEEL?

Door Hans Vedder ● H.Vedder@cimsolutions.nl

Dit verhaal gaat over mijn ervaringen op het gebied van testprofessionalisering. Ik ging werken bij een nieuwe werkgever, waar ik de testorganisatie zou uitbreiden en verbeteren. Uiteindelijk zou het een heel gevecht worden om de testcultuur aan te passen aan de rest van de organisatie. Tevens had ik mijn handen vol met het professionaliseren van de hele ICT. Het zou een ogenschijnlijk onmogelijke klus gaan worden.

Verbeteren en uitbreiden van de testafdeling

De ploeg waarmee ik startte, bestond uit twee internen en twaalf externen. Een maand eerder waren deze mensen gebombardeerd tot testafdeling. Er was geen testbeleid, er waren geen standaards en er was geen eenheid binnen de afdeling. Binnen een jaar lag er een testbeleid, een helder testproces, waren hulpmiddelen beschikbaar en ontstond een hechte groep van testspecialisten. Ondertussen werden alle projecten bemand met één of meerdere testspecialisten, zo goed mogelijk ondersteund vanuit de testafdeling. Min of meer afgedwongen vanuit het management werd er bij elk project een tester ingezet. Deze tester zorgde dan voor het testplan, testdesign, uitvoering en rapportage voor alle testsoorten. Het verbeteren had plaatsgevonden en nu was in mijn ogen de tijd gekomen om uit te breiden. Zowel in activiteiten als in personele aantallen. Ook dat lukte en wierp zijn vruchten af. Bij andere afdelingen ontstond ondertussen interesse waar de testafdeling zich allemaal mee bezig hield.

Na anderhalf jaar stond er een afdeling met gemotiveerde professionals, het testproces was beschreven, standaarden, templates en procedures waren voor handen. En van elke testsoort was wel een mooie gelamineerde grafische weergave aanwezig. Met manschappen en materiaal op orde, kon de aanval op de software beginnen. Het leek een makkie te worden. De testafdeling zorgde voor invulling van het testproces binnen alle projecten en tijdens onderhoud van software en gaf ondersteuning waar dat nodig was.

Kloof tussen test en ICT?

De testprofessionals ondervonden in de projecten dat de buitenwereld niet even gestructureerd was als het eigen testproces. In het begin werd er behoorlijk dwars gelegen als er geen requirements konden worden aangeleverd, of als er bijvoorbeeld in het geheel geen stabiele testbasis aanwezig was. Ook was ik overdonderd door het feit dat een projectleider nog even het testrapport naar zijn hand wilde zetten. Het totale testteam werd verrast door het feit dat ze moesten gaan begrijpen dat er bepaalde basiszaken gewoonweg niet aanwezig waren. Testers vroegen naar zaken die er niet waren. Het werd nog pijnlijker toen uitgelegd moest gaan worden waarom we deze zaken juist zo nodig hebben als testers. Ik heb het hier over zaken als requirements, risicoanalyses, ontwerpen, betrokkenheid, versiebeheer, et cetera.

In feite stond er een testproces dat niet uitvoerbaar bleek te zijn, aangezien de procesverbetering alleen uitgevoerd was bij de testafdeling. Er was een kloof ontstaan tussen de testafdeling en de rest van de ICT-afdelingen (ontwikkel- en beheerafdelingen). En een kloof vraagt om bruggenbouwers. Op het gebied van werkwijzen, kwaliteitsgericht werken en over hoe een software-ontwikkeltraject eigenlijk zou moeten verlopen was de afgelopen twee jaar ➔

weinig tot geen verbetering geweest. Dat er verbeterd kon worden was inmiddels wel bij iedereen helder. Door het management werd een onpartijdige, externe partij gevraagd een CMMi nulmeting uit te voeren, opdat het duidelijk zou worden welke processen we het eerste zouden moeten oppakken om te verbeteren. De testprocessen verificatie en validatie kwamen als beste uit de bus en alle andere processen scoorden bedroevend slecht. De juichstemming bij de testprofessionals duurde niet lang, aangezien de testafdeling geconfronteerd werd met de wet van hun remmende voorsprong.

Aanpassen van de eigen testaanpak

Waar anderen aangaven dat de testafdeling de rest van de organisatie niet begreep, wisten de testers precies hoe de vork in de steel zat. In plaats van de procedures te volgen, gingen ze ervoor zorgen dat anderen beter gingen presteren. Door het aanpassen van de eigen testaanpak werd het makkelijker voor de rest van ICT om aan te haken bij het testproces en andersom. Er ging een wereld voor beheerders open toen de testers hen duidelijk maakten wat zij aan goede acceptatiecriteria kunnen hebben.

Van de standaard testplannen werden afgeleiden gemaakt, waardoor bijvoorbeeld kleinere testtrajecten, zonder te veel overhead aan testproducten, prima geholpen waren. Dit alles zonder afbreuk te doen aan een degelijk testproces. De testprofessionals bleken ook te kunnen scoren op andere gebieden dan alleen maar op testen. Het idee dat de testafdeling ook constructief kon meedenken met de problematiek van andere ICT-afdelingen, werd gewaardeerd. De beeldvorming van wat de testafdeling kon betekenen verbeterde op deze manier. De medewerkers benadrukten dit zelf nog door een manifest op te stellen over de eigen afdeling. Hierin werd aangegeven wat de taken van de testafdeling zijn, waar zij voor op de wereld waren en hoe zij hier mee omgaan. Daardoor benadrukten ze dat we niet de afdeling waren die de testbasis zelf moet maken, maar gaven we aan waar onze grenzen lagen en wat we juist wél als onze taak zagen. Duidelijkheid voorop.

Waardering uit de organisatie

Door uiteindelijk ook nog eens voor meer ambassadeurs te zorgen van de testafdeling ontstond een steeds betere waardering van de testafdeling. Ambassadeurs: tevreden afnemers van onze testfaciliteiten die invloed hadden op moeilijk te bereiken management. In feite werd er niet anders getest dan voorheen, maar nu leek de afstand tussen de testafdeling en de rest van de ICT-afdelingen stukken kleiner geworden. Hierdoor werden de adviezen van de testprofessionals ook veel serieuzer genomen. De testafdeling kwam er langzamerhand achter dat het goed toepassen van de soft skills misschien nog wel meer kon opleveren dan alleen de kennis en kunde van het testproces. Meer dan eens werd het gewaardeerd als de testmanager van een project aanschoof bij de stuurgroep. Testen werd steeds meer gezien als een waardevolle aanvulling op het gehele ontwikkelproces. Niet door verbetering van het eigen proces, maar door het veranderen van de eigen mindset: hoe ervaren anderen het testproces en op welke manier ga je hier mee om?

Conclusie

Door het beter ‘managen’ van mijn eigen omgeving zorgde ik ervoor dat testen op de kaart kwam. Dat hierdoor ook nog eens de hele organisatie professioneler werd, was uiteindelijk een fantastisch resultaat. Het mooiste was wel dat alle testprofessionals van de testafdeling zagen wat ze konden doen en wisten hoe hiermee om te gaan binnen deze organisatie. Sinds die tijd weet ik dat professionaliseren meer is dan alleen een goed team, uitgerust met ➔

allerhande templates en testkennis, neer te zetten. Voor de één zal dat wellicht overkomen als minder professioneel; voor het behaalde resultaat is het uiteindelijk een weloverwogen beslissing geweest. En de ogenschijnlijk moeilijke klus die het zou gaan worden? In werkelijkheid viel het reuze mee. Het lastigste hierin was voor mijzelf om de knop van verbeteren om te zetten naar die van veranderen. Daarbij ook in de gaten houdend met wat voor soort organisatie je te maken hebt, want al begint verbeteren met veranderen, de aanpak is per organisatie weer anders.
←

VISUALIZE YOUR REGRESSION TEST APPROACH

Door Patrice Willemot ● patrice.willemot@ctg.com

Iedere softwaretester weet wat regressie is en weet dat we regressietesten moeten doen, maar hoe komt het dat het zo moeilijk is om de juiste regressie te capteren en daaraan de juiste hoeveelheid testing te koppelen?

Een praktijkvoorbeeld

Laat het me anders uitleggen. Je ben op dit moment bezig met de bouw van een nieuw huis en op een bepaald moment ben je bezig met de badkamer. De loodgieter plaatst een nieuwe douche, een nieuwe lavabo, een nieuw bad en sluit de nodige leidingen aan zodat alles werkt. Daarna komt de elektricien en hij koppelt alle verlichting en stopcontacten. De enige test die de elektricien doet, is een validatie van zijn werk door middel van de verlichting in de badkamer aan te doen. Waarom kan de elektricien dit doen? Hij kan met zekerheid zeggen dat hij geen testen moet doen rond het werk van de loodgieter, omdat er een duidelijk onderscheid is tussen het werk van de loodgieter en het werk van de elektricien.

Waarom is dit niet mogelijk binnen een softwareproject? Waarom moeten we vaak zeer veel regressietesten uitvoeren om enige zekerheid te hebben rond de kwaliteit? De scope van regressietesting is vaak gebaseerd op buikgevoel, ervaring of beschikbare tijd. In een wereld waar de kosten onder druk staan en men zo efficiënt mogelijk te werk moet gaan, moeten we in staat zijn om de juiste hoeveelheid regressietesten te identificeren.

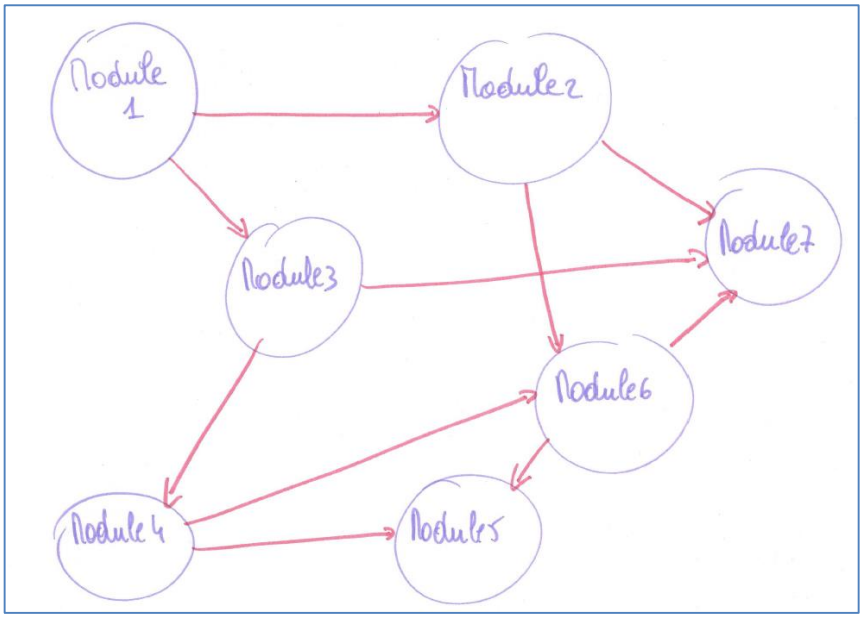
Identificatie regressietesten

De volgende stappen zijn een leidraad om die identificatie te doen.

Stap 1: Weergave van de applicatie die getest moet worden

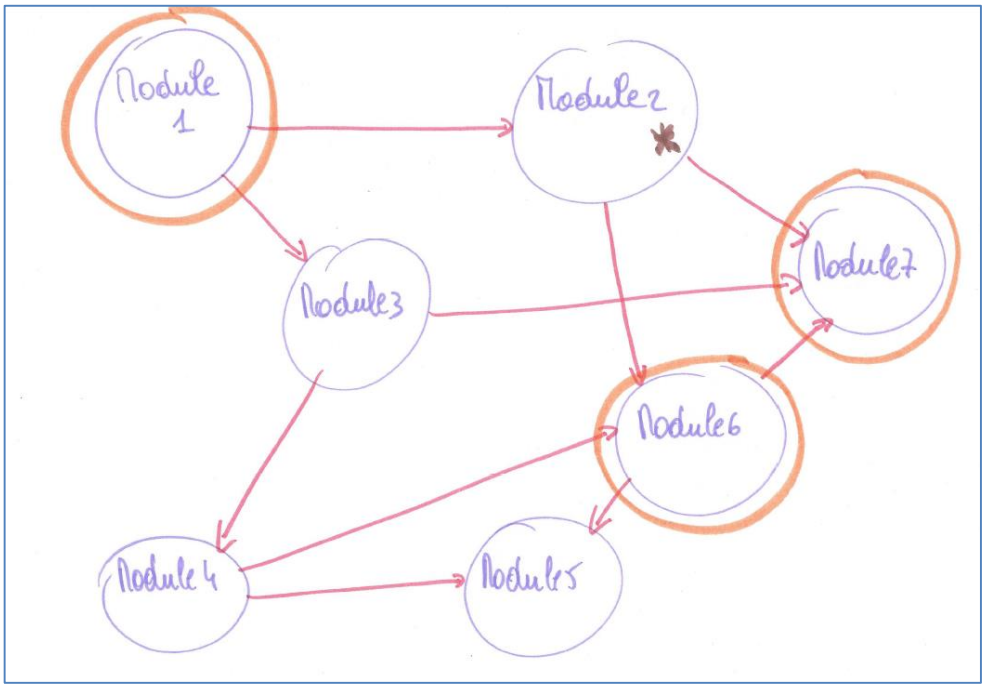
De applicatie goed kennen is een vereiste voordat met regressietesting gestart kan worden. Een gemakkelijke manier om de kennis van een applicatie te hebben, is door een visuele weergave te creëren van de applicatie. Deze weergave is een helikopterzicht van de applicatie, maar met de nodige verbindingen met andere applicaties, systemen of organisaties. Binnen die weergave gaat men de applicatie opsplitsen in verschillende modules. De afhankelijkheden en de datarichtingen worden op deze weergave aangeduid zodat men weet waar de afhankelijkheden en de data flows zijn. Het weergeven van al die afhankelijkheden is zelfs voor een tester met jaren ervaring niet gemakkelijk en de nodige hulp van anderen (projectmanagers, analisten, ontwikkelaars, …) zullen aangesproken moeten worden om deze weergave te creëren en te valideren.

In onze weergave zien we bijvoorbeeld dat Module 6 een uitgaande link heeft met Module 5 en Module 7 en een inkomende link heeft met Module 2 en Module 4. ➔



Stap 2: Identificatie van de wijziging

Wanneer een nieuwe wijziging wordt doorgevoerd op een module binnen de applicatie, kan men dit gemakkelijk aanduiden op de weergave van de applicatie (zie stap 1). Alle gelinkte modules gekoppeld aan de module waar de wijziging heeft plaatsgevonden, kunnen geïmpacteerd zijn door de wijziging. Dit wil zeggen dat we de regressietesten zullen uitvoeren op de module met de wijzing en de daaraan gelinkte modules. Voorwaarde om de juiste module aan te duiden is dat men als tester of als team precies weet waar de wijziging plaatsvindt. Vaak kunnen we dit niet en moeten we (zoals in stap 1) extra hulp krijgen van de andere teamleden. In vele gevallen is het voor hen ook puzzelwerk om de wijziging juist te kunnen aantonen op de weergave van de applicatie.



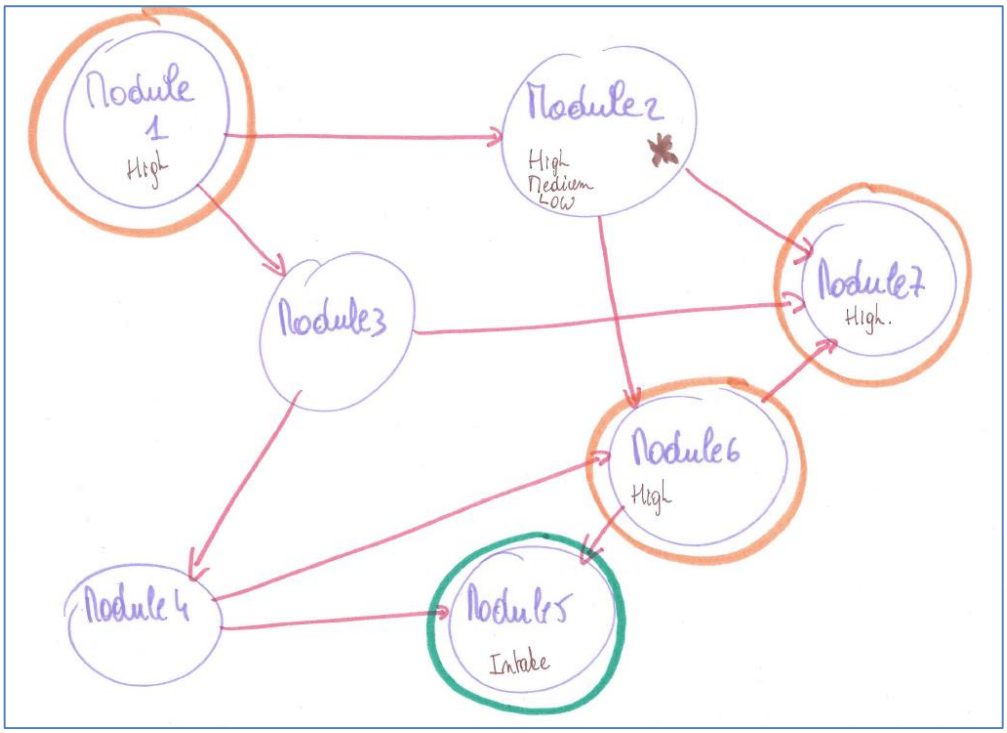
In dit voorbeeld gebeurt de wijzing op Module 2; dit wil zeggen dat de Modules 1, 6 en 7 mogelijks ook geïmpacteerd zijn door de wijziging ➔

Stap 3: Regressietestaanpak

Moeten we alle testcases uitvoeren van alle gelinkte modules? Hebben we hier voldoende tijd voor? Het principe van Risk Based Testing kunnen en moeten we ook toepassen voor het regressietestverhaal.

Een mogelijkheid om Risk Based Testing toe te passen op onze weergave van de applicatie:

1. Voor de module(s) die direct door de wijziging zijn geïmpacteerd, gaan we alle testcases uitvoeren (High, Medium en Low prioriteit). [In het voorbeeld: Module 2]
2. Voor de module(s) die gelinkt zijn aan de module met de wijziging zullen enkel de testcases uitgevoerd van prioritiet High. [In het voorbeeld: Module 6 en Module 7]
3. Voor de modules die gelinkt zijn aan de module met de wijziging door middel van de aanlevering van input, zullen enkel de testcases met prioriteit High uitgevoerd worden. [In het voorbeeld: Module 1]
4. Voor de module(s) die niet direct gelinkt zijn aan de gewijzigde module, maar gelinkt zijn aan de gewijzigde module, hiervoor zullen enkel de intake-testen uitgevoerd worden. [In het voorbeeld: Module 5]



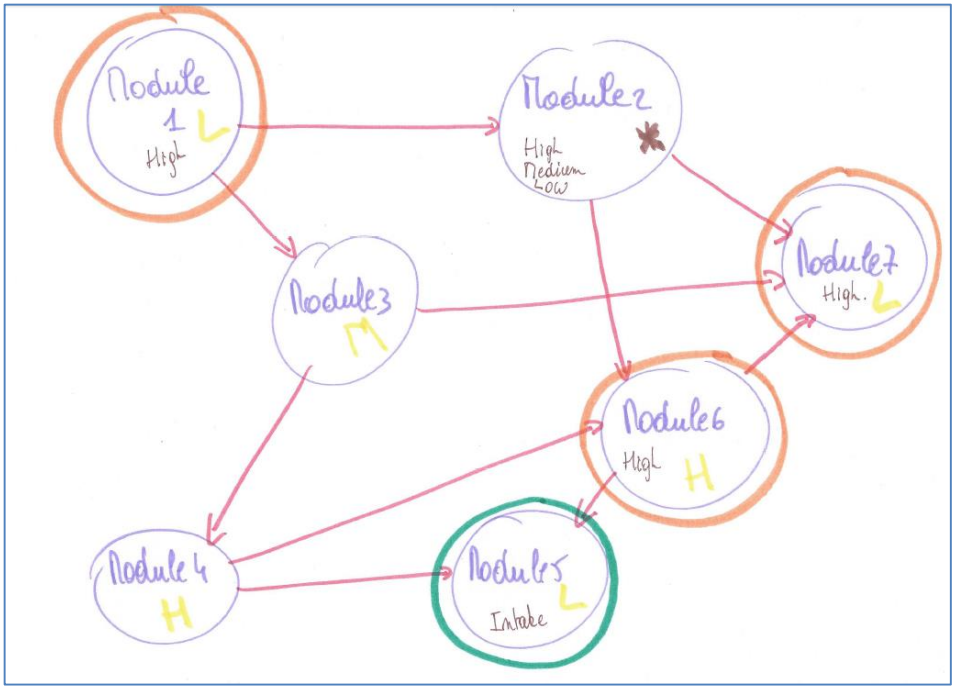
Deze 'Risk Based Testing'-aanpak kan men perfect aanduiden op onze weergave van de applicatie. Voor elke grote wijziging kan het testteam dit aantonen op de weergave, zodat het voor iedereen duidelijk is wat de scope voor regressietesten is.

De aanpak om de regressie te identificeren is oké, maar het kan beter. De afgelopen jaren merken we op dat een tester niet langer een éénzaat is binnen het team - kijk maar naar hoe men binnen een agile team werkt - maar dat ze meer en meer gaan samenwerken met de andere disciplines binnen het team om de krachten bundelen. Het bundelen van die krachten kan en moet ook gebeuren om de juiste regressie testcases te identificeren. ➔

Code complexiteit van de module

Gebaseerd op onze weergave van de applicatie, gecreëerd in de voorgaande stappen, is het mogelijk dat ontwikkelaars een overzicht maken van de complexiteit van de verschillende modules. Is de complexiteit van de module zeer hoog? Bevat deze module veel referenties naar ander modules? Deze informatie kan ons helpen om onze 'Risk Based Testing' aanpak te verfijnen.

In het voorbeeld wordt de complexiteit van de modules weergeven door de gele letter H, M of L. De Modules 6 en 7 hebben een Hoge complexiteit, de Modules 1, 5 en 7 hebben een lage complexiteit. ➔

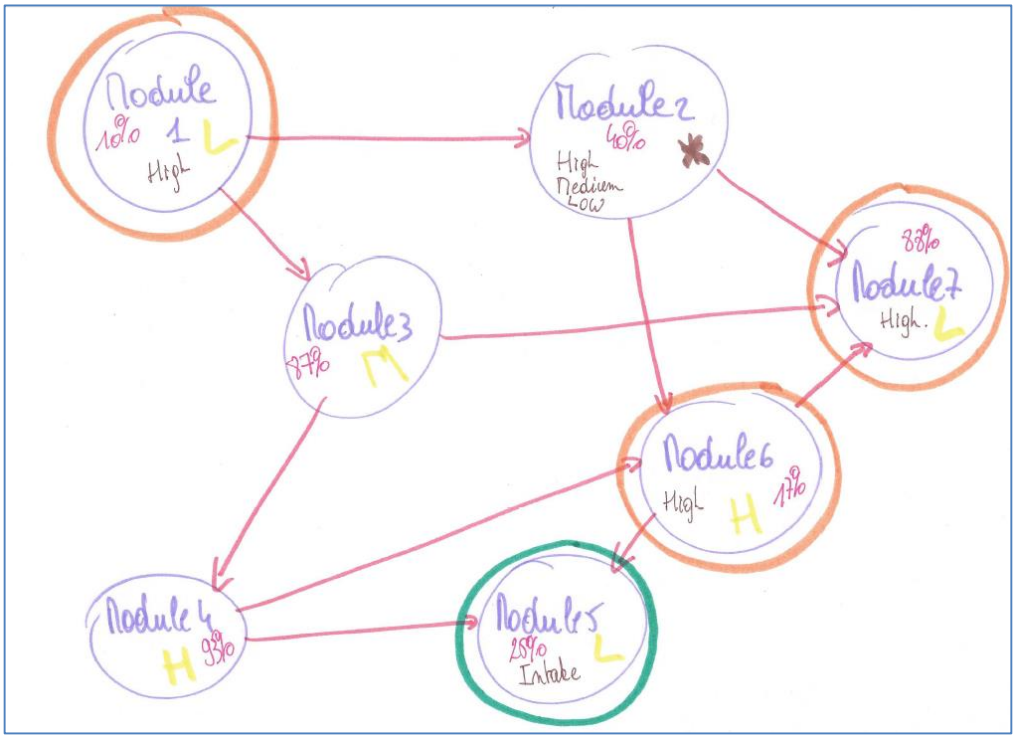


Unit testing

Om onze aanpak nog verder te verfijnen kunnen we gebruikmaken van alle informatie die gegenereerd wordt tijdens de uitvoering van de unit testcases. Zo'n uitvoering geeft niet enkel aan welke delen van de applicatie wel of niet werken, deze uitvoering kan ook aangeven hoeveel unit test coverage er is bereikt per module. Deze unit test coverage zouden we kunnen gebruiken om het aantal testcases binnen de scope van het regressieverhaal te optimaliseren.

% Unit test coverage	Regressieaanpak
Minder dan 20%	High, Medium en Low prioriteit testcases
Tussen 20 en 50%	High en Medium prioriteit testcases
Tussen 50 en 75%	High prioriteit testcases
Meer dan 75%	Enkel intake testcases en een subset van de high prioriteit testcases

In het voorbeeld worden de unit testing coverage aangeduid in het roze. Module 4 heeft de grootste coverage (93%) en Module 1 heeft de laagste coverage (10%).



De combinatie van deze technieken kan je vertalen in volgende aanpak voor regressietesten door middel van een wijziging op Module 2. ➔

Module	Regressietestaanpak
Module 2	**High, Medium en Low prioriteit testcases**
Module 1	Low complexiteit ☐ intake testcases
	10% Unit test coverage ☐ High medium Low testcases
	Conclusie: Intake testcases + high prioriteit testcases
Module 7	Low complexiteit ☐ High testcases
	88% Unit test coverage ☐Intake testcases + subset high prioriteit testcases
	Conclusie: Intake test case + subset High prioriteit test case
Module 6	High complexiteit ☐ High testcases
	17% unit test coverage ☐ High medium Low testcases
	Conclusie: High & Medium prioriteit testcases
Module 5	Low complexiteit ☐ geen testing
	25% unit test coverage ☐ High en Medium testcases
	Conclusie: High prioriteit testcases

Conclusie

Dankzij deze technieken is het mogelijk om je regressietestaanpak weer te geven zonder daarvoor specifieke tools te gebruiken. Niet enkel de aanpak wordt weergeven, maar ook de scope van de regressietesten. Daarnaast kan gemakkelijk aangetoond worden hoe je tot die scope bent gekomen. Het is een vorm van communicatie die weinig inspanning vergt en die hoge impact zal hebben naar andere teams en managers. Elke tester, testcoördinator, testmanager kan deze techniek gebruiken om efficiënter aan regressietesten te doen. ←

'NIET LULLEN, MAAR POETSEN' 2.0

Door Heini Veneberg ● h.veneberg@cimsolutions.nl

Leuk die methoden. Stuk voor stuk bieden ze allemaal voortreffelijke handvatten voor het testvak. Testers waren altijd al zeer betrokken bij, en voelen zich verantwoordelijk voor het goed opleveren van een systeem. Waarom horen we dan toch zo vaak 'Ja, maar..'. Als tester zien we vaak de obstakels al en vergeten we dat we daarmee onze slagkracht (kunnen) verliezen. Op een `Glas is half leeg' zit niemand te wachten, al moet je natuurlijk wel de pijnpunten adresseren. Met andere woorden, gewoon doen. Op zijn Twents 'niet lullen maar poetsen'. Hoe kan de tester door zijn eigen toedoen buiten spel komen te staan? Dit is op zich niet nieuw, daarom ook de toevoeging 2.0.

De gereedschapskist/rugzak

De gereedschapskist/rugzak van een tester is de afgelopen jaren goed gevuld met allerlei gereedschap. Gereedschap in de vorm van methoden en tooling. Dit gereedschap zorgt ervoor dat we als testers eenduidiger kunnen communiceren en rapporteren naar onze omgeving. Dit heeft er mede voor gezorgd dat de testdiscipline geaccepteerd is in organisaties. Het testvak is gegroeid in volwassenheid.

Door de testmethode kunnen we beter aangeven wat we waar en wanneer gaan testen en wat we verwachten van andere disciplines in het aantonen van kwaliteit. Wat verwachten we van de ontwikkelaars op gebied van unittesten? Wat gaan we doen tijdens de systeemtesten? Hoe gaan we de integratietestfase aanpakken? Zodat we uiteindelijk kunnen eindigen in een acceptatietesttraject, waar zo mogelijk de gebruikers mee gaan testen. Dit alles zorgt ervoor dat we een leidraad hebben om continu een uitspraak te kunnen doen over de kwaliteit van het te testen systeem. In onze rugzak hebben we ook diverse manieren om te komen tot een hoeveelheid testcases. Tijd is beperkt, dus we kunnen middels risico gebaseerd testing de juiste testdiepgang weergeven. Hier hebben we aspecten in gebouwd, die we de stakeholders over het algemeen kunnen uitleggen. Moeten we vervolgens testen frequent uitvoeren, dan hebben we inmiddels een ruime gereedschapskist met tooling om testen te automatiseren. Vooral in omgevingen met continuous integration/ -delivery kunnen we `snel` een testantwoord geven. In het bijzonder als je in een Agile omgeving moet testen, waar je regressie sneller groeit dan in een traditionele waterval omgeving. Methode en tooling zijn op orde, zou je kunnen stellen! Het zal meegroeien/evolueren met de andere ICT-veranderingen.

Waarom loopt het dan nog regelmatig spaak? Of iets positiever: Waarom loopt het soms even niet zo soepel?

Extra gen

De tester heeft ook menselijke trekjes. Persoonlijk denk ik dat de tester nog een extra gen heeft. Dat is zijn kwaliteitsbesef. Daarmee zeg ik niet dat de anderen dat niet hebben. Nee, de andere disciplines gaan anders met dat besef om. ➔

Zelf leg ik dit altijd als volgt uit:

Situatie:

Via businessanalyse en informatieanalyse is er een mooi document gemaakt met alle requirements. Echter, als je er goed naar kijkt, dan is het inconsistent of onduidelijk en roept het allerlei vragen op.

Hoe gaat de ontwikkelaar hiermee om:

De ontwikkelaar zal vanuit zijn expertise deze inconsistentie/onduidelijkheid bespreken met de business. De business verschaft andere/aanvullende informatie. De ontwikkelaar zal na een aantal pogingen zijn betoog staken. De ontwikkelaar kiest de strategie `ik ontwikkel het gewoon, zoals ik denk dat ze het willen hebben. Willen ze het anders, dan hoor ik het wel`.

Hoe gaat de tester hiermee om:

De tester zal vanuit zijn expertise deze inconsistentie of onduidelijkheid bespreken met de business. De business verschaft andere informatie. De tester blijft langer bezig met het helder krijgen van de requirements, aangezien zijn taak is uiteindelijk een uitspraak te doen over de kwaliteit of dit het systeem is wat de business wil hebben.

Door dit kwaliteitsbesef-gen zien we veel pijnpunten in organisaties en projecten, die het lastig maken om het testtraject ideaal te laten verlopen. Met andere woorden, de `Ja, maar' (betekent eigenlijk 'Nee') ligt weer op het puntje van onze tong. We hebben het glas alweer half leeg.

Zelfbeeld verbeteren

Oké, we kunnen dus aan ons zelfbeeld werken of anders het beeld dat anderen van testers hebben of krijgen. Moeten we dan onze taak verloochenen? Nee!

Hieronder zal ik een aantal situaties schetsen, waar de omgeving de constateringen van ons testers vervelend vindt. Waardoor we het beeld geven van `Ja, maar` of `zie je nu wel`! Soms merk ik eigenlijk op dat de omgeving je wel gelijk wil geven, maar het komt op zo'n vervelend moment en heeft men al genoeg aan hun eigen uitdagingen en dan komen wij nog even met wat extra sores. Kunnen we het als testers dan wel ooit goed doen? Natúúrlijk, we staan ergens voor.

Onduidelijke requirements! Dit probleem is gelukkig door de introductie van Agile werken al aanzienlijk afgenomen. De methode zorgt ervoor dat de brokjes beheersbaar zijn geworden. Hebben we dan nog onduidelijkheden, dan kunnen we tijdens de stand-up sessie hier melding van maken. Als je dan een Scrum-master hebt die weet hoe de methode werkt, dan wordt er vervolgens gefaciliteerd dat je het aan de orde kunt stellen en voor een oplossing zorgen. De tester moet gewoon zijn thema of vraag goed brengen en dan zal de organisatie moeten faciliteren. Komt hier een `ja, maar` gevoel naar boven, dan is de Agile methode niet perfect qua uitvoering of de wijze van brengen van de boodschap is onduidelijk. Als tester moet je natuurlijk wel in staat zijn niet jouw probleem te vertellen, maar aangeven wat het gevolg is voor het te ontwikkelen systeem.

Leveringen van onvoldoende kwaliteit! Je kent het wel, de build zou op dag X komen en je krappe testtijd verdampt waar je bij staat, gezien het feit dat je dat je pas kunt beginnen te testen op dag X+tig dagen. Heerlijk die continuous integration/delivery oplossingen. Zorgt voor een perfecte delivery straat, zodat je als tester altijd op tijd kunt beginnen met je testuitvoering. Ho stop, even terug naar de realiteit. Zelfs met deze oplossingen loop je tegen obstakels aan. Nu kun je op het moment dat je dit constateert weer gaan klagen! Hier zit heel vaak de oplossing in de wijze van afspraken maken vooraf. ➔

Welke bedoel ik dan:

- 'Definition-of-Done' goed definiëren; eigenlijk zegt de term op zich al voldoende. Klaar betekent niet alleen komt door de compiler, maar bijvoorbeeld de build moet installeerbaar zijn op het testsysteem. Met andere woorden, als tester heb je kans of dit aan te scherpen vooraf en tijdens de retrospective kun je invloed uitoefenen als dit onvoldoende toegepast wordt of zelfs aanscherpen wanneer nodig.
- Gebaksconstructie; als iemand zich niet goed gehouden heeft aan bijvoorbeeld de 'Definition-of-Done', dan wordt er getrakteerd door de boosdoener. Dit zorgt ervoor dat iedereen zich goed bewust is van de spelregels. Het constateren wie, hoeft niet alleen door de testers te gebeuren. Als de `Definition-of-Done`goed is ingeregeld, zal het team dit zelf wel constateren. Als tester kun je dan ook spontaan eens trakteren als je de zoveelste build meteen kon gaan testen. Met andere woorden, soms moet je het team complimenteren dat het loopt zoals we het hebben afgesproken.

- Wijze van communiceren mocht het dan toch een keer niet werken. Wees ervan bewust dat waar gewerkt wordt, spaanders vallen. Meld de situatie op een positieve en constructieve wijze, maar ook weer niet te lief. Waarbij je wel netjes aangeeft wat de gevolgen zijn voor het testen.
- Bevindingen worden `gereject` zonder overleg! Als dit gebeurt, dan levert dit vaak veel frustratie bij de indiener op. Dit zorgt vervolgens voor veel communicatie, waarbij we ons als testers uiten op een wijze die door de omgeving als onprettig wordt ervaren. Wij zijn dan de dwarsliggers enzovoort en begrijpen dan niet dat deze bevinding totaal niet belangrijk is. Hoe zorg je nu dat dit proces in goede harmonie gaat verlopen? Dit betekent gewoon vooraf een proces afspreken. De meesten zeggen dan: daar heb je een CCB voor. Echter, als je een groot project hebt, dan zit daar een afgevaardigde van de testers, die echt niet alle bevindingen in detail kent. Ik ben voorstander van het volgende: als een bevinding door een behandelaar (meestal ontwikkelaar)`gereject` moet worden, dan is er tussen de behandelaar en de indiener afstemming geweest. Als ze gezamenlijk er over eens zijn, dan kan een reject plaatsvinden.
- 'Lastige en complexe bugs'! Je kunt gewoon een bevinding aanmaken en afwachten. De kans op een ping-ponggedrag van bevindingen is hoog. Dit levert veel frustraties op. Niet alleen bij de indiener, maar zeker bij meer mensen in het project. De oplossing is: haal er een ontwikkelaar uit en vraag om hulp met als verzoek 'Ik snap het gedrag van het systeem niet helemaal'. Negentig procent van de gevallen zal tijdens de uitleg qua werking men gezamenlijk tot de conclusie komen dat het een bug is. Scheelt veel frustraties en tijd.

- Nu schijnt de loopafstand van tester naar behandelaar kleiner te zijn dan die van behandelaar naar tester. Hiervoor heb ik altijd het instrument snoeppot toegepast. Door te zorgen dat de behandelaar sneller langs komt, moet er wat te halen (lees: snoep) zijn. Dit werkt echt!

Transparantie

Eigenlijk ga ik zoveel mogelijk op zoek naar wat wel duidelijk is en om van daar uit de onduidelijkheden te adresseren en die van een oplossing te voorzien. De testaanpak transparant maken en laten zien hoe en wat onze aanpak is. Nee, niet alleen een testplan opleveren. Ook op de verschillende momenten laten zien, wat we toevoegen. De methoden bieden een mooie basis, de rest zit dus in de wijze waarop je vertrouwen krijgt. Waarbij het ook belangrijk is om voordat je in de kritische fase zit, eigenlijk al met elkaar in gesprek bent over de zachte aspecten van het ➔

project. Hoe en wat je van elkaar verwacht in de samenwerking en wat je van elkaar kunt verwachten. Gedurende de kritische momenten is het vervolgens van belang dat je let op de wijze van je communicatie. De wijze van communiceren zal zeker van invloed zijn op de reactie van het project. Waarbij een tester zeker zwaar moet inzetten, als cruciale zaken niet goed werken. Echter, ook dit moet je brengen als een uitdaging voor het hele team. Dus de soft-skill antenne van een tester moet goed ontwikkeld zijn. Daarbij kan een tester veel krediet opbouwen door niet alleen de problemen te melden.

Door ook na te denken over een voorstel tot oplossing zorg je voor geloof en vertrouwen dat het systeem goed opgeleverd kan worden. Hieronder twee voorbeelden, de eerste een mindere aanpak en de tweede het juiste voorbeeld:

1. Ik geef een negatief advies met betrekking tot het gaan richting productie.
2. Ik kan een positief advies geven om in productie te gaan als we de volgende bevindingen nog even oplossen.

Vanuit alledaagse voorbeelden maak ik gebruik van simpele instrumenten, zodat de omgeving vertrouwen krijgt in de aanpak. Als het vertrouwen en geloof in de aanpak er is, dan is het project al voor de helft geslaagd. Middels de diverse methoden, technieken en tools kun je vooraf goede afspraken maken, waardoor je goed van elkaar weet hoe we met elkaar omgaan. Vervolgens kun je op de momenten dat het erom gaat letten op je soft skills.

Dus mouwen opstropen en aan de slag! Aan de slag met de juiste vorm van communicatie. Daarom `niet lullen, maar poetsen` 2.0. ←

DE MYTHE VAN DE TIJDBESPARING OP TESTUITVOERING

Door Erik Jansen ● Erik.Jansen@bartosz.nl

Afgelopen zomer vond er een grote planningsworkshop plaats op het project waar ik testmanager ben. Deze workshop zou precies in de eerste week van mijn zomervakantie plaatsvinden. Nu heb ik hart voor de zaak, maar hecht ik ook veel waarde aan mijn zomervakantie. Dus heb ik voor mijn zomervakantie aanbevelingen achtergelaten, waarop ik het nu volgende artikel heb gebaseerd.

De testuitvoering

Zodra alle ontwerp- en bouwactiviteiten zijn afgerond, start de meest onvoorspelbare fase van het project, de testuitvoering. Alle defects zijn geïntroduceerd en nu is het aan ons, testers, er zoveel mogelijk te vinden. Voor de start van de testuitvoering weten we niet hoeveel het er zijn, wanneer we ze zullen vinden en hoeveel tijd het gaat kosten om ze op te lossen. Natuurlijk is het niet noodzakelijk dat we alle defects vinden, maar zelfs een garantie dat we de meest ernstige vinden, kunnen we niet geven. Elke voorspelling kan de juiste zijn; ga dat maar eens plannen. In feite kunnen we sinds de invoering van Risk Based Testen binnen elk gegeven tijdsspanne een testrapport opleveren. In theorie zelfs na een uurtje, al is dat meestal niet het bericht waar de stakeholders van het project op zitten te wachten. Een goede aanpak voor een testuitvoering, gebaseerd op een rijke verzameling aan projecten, en bijbehorende mislukkingen, is de volgende:

- Een periode inplannen van bijvoorbeeld drie weken om een eerste indruk te krijgen. Doel is om alle geplande testgevallen minimaal éénmaal uit te voeren. In deze fase gaan nog allerlei basale dingen mis; applicaties die nog niet beschikbaar zijn, login id's die niet werken, omgevingen die nog niet gekoppeld zijn, enzovoort. Neem voor deze fase de meeste tijd omdat je hier de grootste problemen tegenkomt. Omdat je deze problemen meestal niet zelf kunt oplossen, kosten ze vaak veel tijd.
- Een periode inplannen van bijvoorbeeld twee weken om alle defects te hertesten, die gevonden zijn in de eerste fase en ook zijn opgelost. In de praktijk zal het oplossen en testen van bevindingen wat door elkaar lopen; dat geeft niet.
- Een periode van nog eens drie weken inplannen om een 'clean run' uit te voeren. Vanaf de start van deze periode mogen er geen aanpassingen op het systeem meer worden uitgevoerd, om regressie te voorkomen.

Het voordeel van de clean run

Die laatste periode, en vooral het aspect van de 'clean run' is belangrijk. We kennen allemaal wel voorbeelden van waar dat mis kan gaan. Zo zal ik een voorbeeld geven. Eén uit mijn eigen historie, uit de tijd van de sequentiële tapebestanden. Voor een project moeten de jaaropgaven van twee miljoen klanten de adresgegevens worden samengevoegd met de belastinggegevens. Alles was getest en liep goed. Er was echter een klein probleem; er miste een gegeven bij de belastinggegevens. Oké, dit leek geen probleem. Bestand bijgewerkt, kijken of de gegevens er stonden en hup, naar de drukker. Alleen waren de twee bestanden nu niet meer in dezelfde volgorde gesorteerd; namelijk één op postcode en één op sofinummer. Tja, dat levert op twee miljoen records ongeveer 1500 hits . Resultaat 1500 goede jaaropgaven in plaats van twee miljoen uit de printstraat van de drukker; een dure fout.

Die 'clean run' is dus belangrijk en wordt vaak niet voldoende serieus genomen. Ik geef de opdrachtgevers altijd aan dat het oplossen van een fout ook een change is. Alleen één die onder grotere druk tot stand komt. Daarom is ➔

een laatste testperiode zonder foutoplossing nodig. Geef dan ook duidelijk van tevoren aan, wat dat betekent. Als er toch een fout wordt gevonden in de laatste testperiode, betekent dat:

1. óf dat de fout geaccepteerd wordt;
2. óf dat er een volgende periode van herstel en (regressie)test nodig is.

Het eerste punt kan in sommige gevallen wel eens de voorkeuroptie zijn. Gelukkig hebben we tegenwoordig vaak een Product Risico Analyse (PRA) uitgevoerd, zodat de afspraken die daar gemaakt zijn, waarde aan de discussie kan bijdragen. Een goed uitgevoerde PRA kan zeker bijdragen aan het nemen van de juiste beslissing. Bij projecten in problemen komen de geplande testperioden wel eens onder druk te staan. Dan wordt gevraagd waarom de software delivery niet in vier in plaats van zeven weken opgeleverd kan worden. Natuurlijk, je kunt ook alleen naar rechts kijken als je de straat oversteekt; dat kan vaak goed gaan. Echter, dit is niet iets dat ik adviseer. Wat dat betreft vergelijk ik testuitvoering wel eens met een berg zand. Je kunt er wat afhalen, maar het is nog steeds een berg zand. Daar kun je zelfs even mee doorgaan, maar wanneer houdt het op een berg zand te zijn? Dus op de vraag of ik in minder tijd kan testen is het antwoord meestal: 'Ja, maar weet je zeker dat je dat wilt?'. Over het algemeen hebben we geaccepteerd dat we niet alle fouten vinden. Hoe meer je test, hoe meer je er vindt. In elk geval: hoe minder je test, hoe meer fouten je achterlaat en je hebt geen idee hoeveel en hoe ernstig ze kunnen zijn.

Combineren van testsoorten een goed plan?

Nog zoiets, kunnen we niet parallel gaan testen? Bijvoorbeeld de SIT met de UAT combineren, dan winnen we ook tijd. Goed idee, dan wachten er twee teams totdat de SIT defects zijn opgelost in plaats van één team. De meeste tijd wordt echter niet besteed in de uitvoering van testgevallen, maar in het vinden van de oorzaak en vervolgens het beleggen van de defects bij een oplossende partij. Daarnaast moet ook de nodige aandacht worden besteed aan alle coördinerende activiteiten eromheen. Het wordt dan alleen maar complexer als er twee teams parallel testen. In de praktijk loopt de testuitvoering dus gewoon uit de planning bij parallel testen. Het netto-effect is soms dat het meer tijd kost en in elk geval meer frustratie oplevert.

Beter plannen van de testuitvoering

In de meeste gevallen kunnen we best een kortere testuitvoering plannen. Maar is het testen dan ook eerder afgelopen? Kunnen we daarop sturen? Zullen we fouten eerder vinden en ook eerder oplossen? Het is mogelijk, maar je kunt het niet plannen. Als we van tevoren wisten welke fouten we zullen gaan vinden, dan was de noodzaak van testen er niet. Dan zouden we dat van tevoren de defects aan de ontwikkelaar vertellen en zou iedereen blij zijn. Helaas kunnen we niet voorspellen. Wat wel mogelijk is, is om het aantal te vinden fouten te voorspellen. Op een project waarop ik werkte, ontspon zich een discussie met de projectmanager van de ontwikkelaars over de planning. Elke week dat ik de planning inleverde van de testperiode, betekende een extra week ontwikkeltijd voor zijn team en vice versa. Ik kreeg de vraag of ik die tweede run ook in één week kon uitvoeren. Nu had ik in een eerdere release ruim honderd fouten gevonden en deze release was vergelijkbaar qua grootte en functionaliteit. Dus mijn antwoord was: 'John, ik ga 100 fouten vinden, 25 in elke van de eerste drie weken. Hoe snel kun jij ze opgelost hebben?' Dat was gebaseerd op de ervaring van de eerste release.

De duur van de testuitvoering wordt slechts gedeeltelijk bepaald door het aantal testgevallen. Een grotere invloed op de testuitvoering hebben echter de zaken eromheen zoals: defect management, omgevingbeheer, testers begeleiden (bij UAT) en wijzigingen doorvoeren. ➔

Agile versus Waterval

Hoewel tegenwoordig steeds meer Agile wordt ontwikkeld, gebruiken nog steeds veel projecten de klassieke watervalmethode. Maar ook in een Agile omgeving zal dit probleem zich, met kortere tijdlijnen voordoen. Bijvoorbeeld na oplevering van een aantal sprints zal de behoefte om een uitgebreidere regressietest toenemen, met bijbehorende langere doorlooptijd. Agile is relatief nieuw en zo ook de ervaringen daarmee. Tijdens het voorjaarsevenement ga ik graag met het publiek de discussie aan omtrent Agile versus Waterval.

Tot slot

Varianten op de bovenstaande beweringen en meningen gebruik ik al jaren in alle projecten. Over het algemeen met succes. Mocht je na het lezen van dit artikel nieuwsgierig zijn geworden naar meer praktijkverhalen over de boeiendste fase van het testen, namelijk die van de uitvoering, dan nodig ik je uit voor mijn presentatie op het voorjaarsevenement van TestNet met de titel: 'Risk Based Testuitvoering, de weerbarstige praktijk'.

Ik sluit graag af met de volgende aanbeveling naar de lezer:

I hope you will consider above in your replanning sessions before you cut on test execution time. And find a way to get back to 8 weeks. In full confidence of wise decisions. ←

FAST FORWARD MET FITNESSE: TESTAUTOMATISERING IN EEN AGILE BI-PROJECT

Iris Groenewoudt (links) ● iris.groenewoudt@ordina.nl
Marieke Roelofs (rechts) ● marieke.roelofs@ordina.nl


Business Intelligence is een vakgebied dat testen nooit zo veel prioriteit heeft gegeven. Dat is nu aan het veranderen. Zeker nu er steeds meer Agile BI-projecten worden uitgevoerd, wordt testen, en dan het liefst geautomatiseerd, steeds belangrijker. Het afgelopen jaar zijn wij als testers vanuit Ordina betrokken bij een Agile project dat voor een grote zorgverzekeraar een nieuwe BI-omgeving neer zet. Niet alleen dát is een hele onderneming, het inrichten van het testproces is dat ook. Welke stappen zijn er het afgelopen jaar gezet bij inrichten van het automatisch testen en het inbedden hiervan in het ontwikkelproces?

Maart 2013: frustratie alom

Herkent u dit: tientallen handelingen zijn nodig om de uitgangssituatie voor de test klaar te zetten. Als een van deze handelingen vergeten wordt, of in de verkeerde volgorde wordt uitgevoerd, kun je weer helemaal opnieuw beginnen. Helaas kom je hier vaak pas achter als de test niet het resultaat geeft dat wordt verwacht. Tijdverlies en gefrustreerde testers zijn het resultaat.

Zo zag onze maandelijkse regressietest eruit voordat we gingen automatiseren. Niet alleen frustreerde dit repetitieve, handmatige werk ons zeer, ook konden wij niet altijd helemaal instaan voor de betrouwbaarheid van de tests. Helemaal lekker voelde het niet.

Sinds januari hadden we de beschikking over TestFlow, een tool voor geautomatiseerd BI-testen. Maar doordat we zo druk waren met bovengenoemde repetitieve handelingen, kwamen we er niet aan toe de tool te leren gebruiken. Laat staan dat we konden gaan nadenken over een handige inrichting ervan. Zoals dat zo vaak gaat: de waan van de dag weerhield ons van lange-termijninvesteringen.

TestFlow is een uitbreiding van de open source tooling Fitnesse/DbFit. In TestFlow is code toegevoegd die het mogelijk maakt om ETL flows vanaf de command line te starten.
Fitnesse (TestFlow) maakt gebruikt van wiki pagina's waarop tests gedefinieerd kunnen worden. Inclusief de voorbereiding die nodig is om de test uit te kunnen voeren. Deze pagina's kunnen worden samen gevoegd tot test suites. Bij het uitvoeren van een test of suite worden de resultaten in een gekleurde balk boven in de pagina weergegeven.

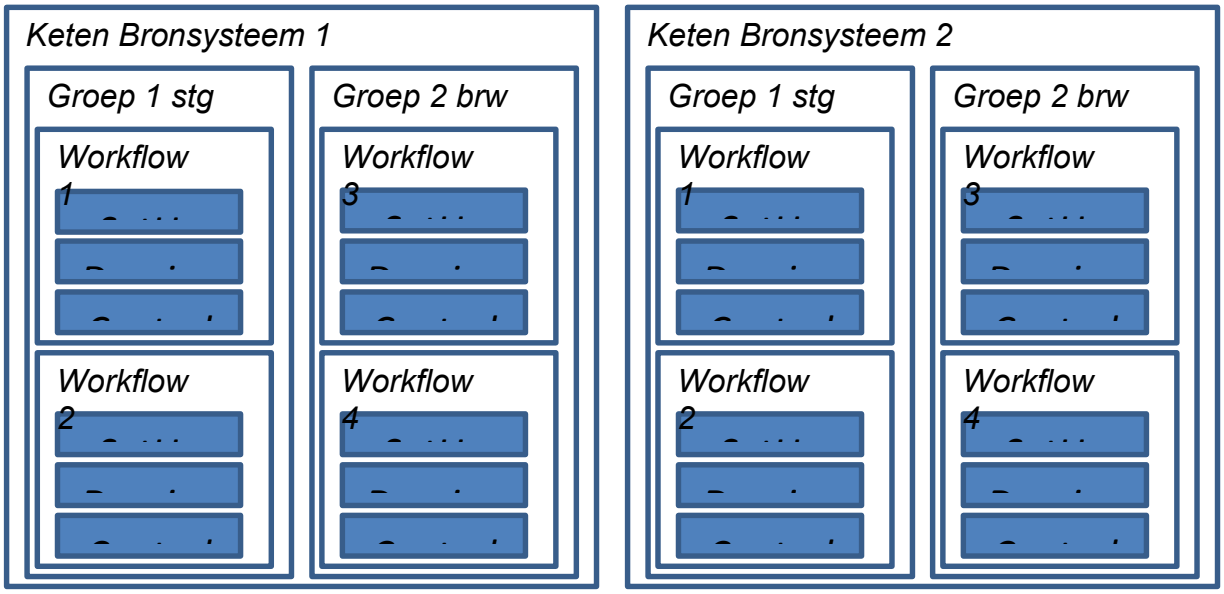
April 2013: het moet er nu toch eens van komen

We besloten dat het zo niet langer kon en roosterden onszelf een paar middagen uit om de tool in te richten. Toen we eenmaal op gang waren, waren we er zo op gebrand om bepaalde dingen werkend te krijgen, dat we ook binnen de gewone 'waan van de dag' elk vrij moment pakten om verder te werken aan de tool. Met als eerste resultaat …

Mei 2013: geautomatiseerde regressietest op de testomgeving

Een ware mijlpaal: de in de eerste alinea opgesomde handelingen konden we voortaan met een druk op de knop uitvoeren. Halleluja! Voortaan was onze maandelijkse regressietest een kwestie van aanzetten en controleren of alle tests op 'groen' stonden. Hierdoor hielden we genoeg tijd over om op onderzoek uit te gaan wanneer de tool op ➔

een bepaald onderdeel geen 'OK' gaf. Door gebruik te maken van generieke componenten zoals de set-up, draaien en controleren pagina's en door het handig groeperen van pagina's (zie figuur 1) is de regressietest eenvoudig onderhoudbaar en uitbreidbaar. Belangrijk, want we werken Agile en elke maand wordt er dus nieuwe programmatuur opgeleverd.



Figuur 1: Ketens per bronsysteem die met behulp van parameters gestuurd kunnen worden en die gebruikmaken van generieke componenten zoals de set-up, draaien en controleren pagina's

Augustus 2013: unittest frameworks ontwikkeld

En dan is het natuurlijk ook wel handig als elke ontwikkelaar dezelfde scenario's draait in de unittest. Dat vonden onze ontwikkelaars ook, en die mooie tool van ons, die zagen zij ook wel zitten. In onderling overleg zijn er templates gemaakt, voor elk type tabel een. Nieuwe programmatuur die eenzelfde soort tabel vult, kan zo heel snel en eenvoudig voorzien worden van een bijbehorende unittest. Een unittest die bewaard blijft en daardoor herhaald kan worden op andere omgevingen, en herhaald kan worden op de ontwikkelomgeving wanneer er aan de betreffende programmatuur is gesleuteld.

De keten per bronsysteem wordt met behulp van een link in een testsuite opgenomen. Dit kan op het niveau van het bronsysteem zelf, of elk niveau daaronder.
Het is ook heel eenvoudig een testsuite samen te stellen waarin bijvoorbeeld links naar de staging workflows van alle bronketens zijn opgenomen. Testpagina's kunnen zo in meerdere testsuites gebruikt worden en toch maar op één plek onderhouden worden.

November 2013: de 'nachtelijke run' op de ontwikkelomgeving

Als we dan toch op de ontwikkelomgeving bezig zijn, en we hebben op de testomgeving een testsuite die alle tot nu toe gebouwde programmatuur in de juiste volgorde aftrapt en de resultaten controleert, waarom zouden we die dan eigenlijk niet ook op ontwikkelomgeving gebruiken? Op die manier kan al het bouwwerk dat op een bepaalde dag is ingecheckt, vrijwel meteen in samenhang met de overige workflows getest worden.

Dankzij deze 'nachtelijke run' komen belangrijke fouten in (combinaties van) software binnen een dag aan het licht. Natuurlijk zijn voor minder voorspelbare fouten aanvullende tests nodig. Daar hebben wij als testers nu ruim de tijd voor dankzij de automatisering van de nachtelijke run. En alle query's die we bij ons testwerk construeren, nemen we op in de tool, zodat ze steeds opnieuw kunnen worden gedraaid, op de verschillende omgevingen. ➔

December 2013: rapportages met een eigen wil

Hebben we nu dan het walhalla bereikt? Not quite. Bovenstaand verhaal gaat geheel en al over ETL-programmatuur. Maar ons project levert ook OBIEE-rapportages op en de deployment hiervan kent wat haken en ogen. Na overzetten van de ene omgeving naar de andere ontdekten we regelmatig dat een bepaald rapport spontaan een fout gaf die er op de vorige omgeving nog niet was geweest.

Hadden we zoveel ETL-tests geautomatiseerd, moesten we alsnog repetitief en handmatig alle tabbladen en drills in de rapporten aan klikken om te kijken of alles nog werkte. Jammer. Dat vonden ook onze OBIEE-ontwikkelaars, die zich, geïnspireerd door de positieve ervaringen van de ETL-ontwikkelaars, ook maar eens op de tool gingen storten.

Februari 2014: technische controle OBIEE-rapportages geautomatiseerd

Met zogeheten 'agents' kunnen we de rapporten automatisch verversen. Wanneer een object niet gevonden kan worden, of er is iets anders mis, volgt er een foutmelding. Met behulp van de tool kunnen we al deze agents achter elkaar aftrappen en de resultaten controleren. Hierdoor hoeven wij ons niet meer bezig te houden met de technische checks en kunnen ons richten op inhoudelijke controles.

Toekomstplannen

Dit alles smaakt natuurlijk naar meer. Als we heel eenvoudig de tests kunnen uitvoeren op zowel de ontwikkel- als de testomgeving, zouden we ze dan voortaan niet ook mee opleveren met de programmatuur? Dan kunnen ook de beheerders er in de acceptatie- en eventueel productieomgeving hun voordeel mee doen.

En zou het niet mooi zijn als er automatisch een test wordt afgetrapt wanneer iemand nieuwe OBIEE-programmatuur incheckt?

En als we een en ander nog flexibeler en onderhoudbaarder inrichten dan nu al het geval is?

En als we kunnen loggen welke versie van de programmatuur de tool heeft gedraaid?

Kortom, de mogelijkheden zijn nog lang niet uitgeput. Maar terugkijkend op waar we een jaar geleden stonden, hebben we een enorme kwaliteitsslag gemaakt. De producten die ons team oplevert, hebben steevast een hoge basiskwaliteit. We hoeven minder te documenteren, omdat de tool de testresultaten bewaart. En doordat zowel ontwikkelaars als testers minder tijd kwijt zijn aan handmatige, repetitieve handelingen, kunnen zij zich op interessantere dingen richten en zich verder ontwikkelen.

TestFlow is allang geen speeltje van de twee testers meer; het heeft zich ontwikkeld tot een onmisbare steun voor het hele team. Het vormt als het ware de ruggengraat van het project. We kunnen niet meer zonder, en we willen niet meer zonder.

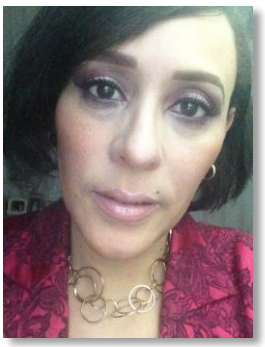
Voorjaarsevenement TestNet

In onze presentatie op het Testnet Voorjaarsevenement laten we onder andere zien wat wij tot nu toe hebben bedacht om de tool zo flexibel en onderhoudbaar mogelijk in te richten. Hoe ziet een dergelijke keten er dan uit? Hoe zorg je dat je afzonderlijke onderdelen ook los kunt draaien? En dat je een wijziging in bijvoorbeeld een tabelnaam niet op zes verschillende plaatsen apart moet doorvoeren?

Verder geven we natuurlijk een live demo van de tool in actie. ←

'SCRUM IN NAME ONLY' – ZIJN WE ECHT VOLGENS SCRUM AAN HET ONTWIKKELEN?

Door Lysette Caetano do Rego – Richardson ● l.Caetano.do.Rego@cimsolutions.nl

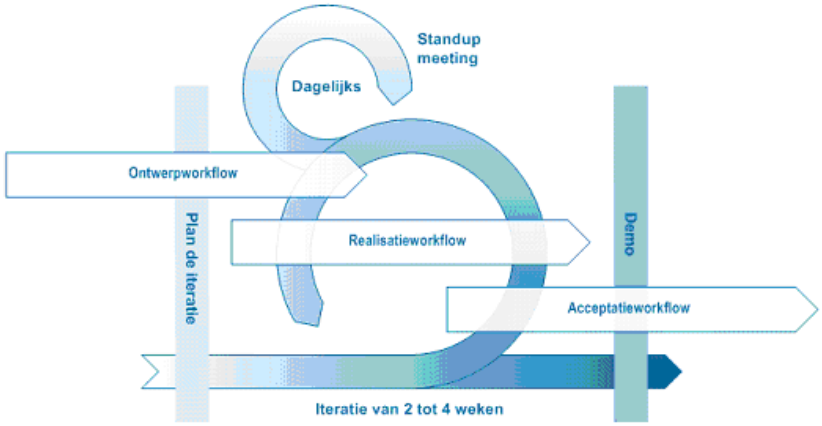


In de bijna vijftien jaar dat ik als tester werkzaam ben, heb ik grotendeels in waterval ontwikkelprojecten gewerkt. Niets nieuws hier, het merendeel van alle ICT-professionals heeft en werkt nog steeds in een waterval ontwikkelomgeving. In mijn verhaal ben ik uitgegaan van een basiskennis van SDM. Hierdoor hebben velen, wat ik noem, het waterval jasje aan. Met andere woorden, de watervalmethodiek is de mindset van waaruit wordt gehandeld. Hier is niets mis mee, vind ik, totdat je in een project komt waar met de Scrum-methodiek moet worden gewerkt.

In 2008 heb ik voor het eerst kennis gemaakt met Scrum. Ik kwam in een heel leuk project terecht met een vooruitstrevende projectleider. De website waaraan ik heb gewerkt, werd – en wordt nog steeds – als proeftuin gebruikt om nieuwe methodieken uit te proberen. Tot op dat moment wist ik niets van Scrum of Agile en wat ik altijd doe wanneer ik iets nieuws tegenkom, is mij er helemaal in verdiepen. Er was een kast met allerlei leesmateriaal over Scrum en Agile en ik heb ze allemaal doorgenomen. Ik heb ook de cursus Agile Testing gedaan om te leren hoe te testen in een project dat de Scrum-methodiek toepast. Ik heb dus een poging gedaan om mijn waterval jasje uit te trekken en het Scrum jasje te proberen.

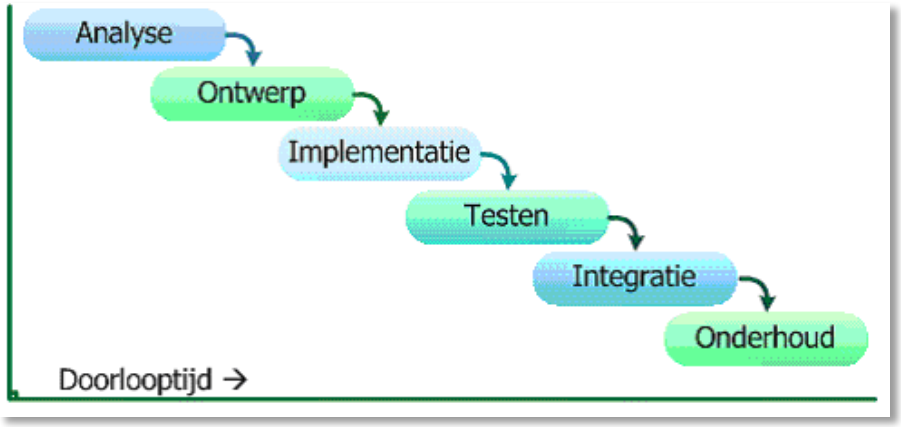
Figuur 1. Scrum jasje uitproberen

Dit laatste dacht ik tenminste dat ik aan het doen was. Iedereen die kennis van Scrum heeft, weet dat je binnen deze methodiek met korte sprints werkt. Daarbij werk je in een multidisciplinair team samen waarbij de product owner bepaalt wat er binnen de iteratie wordt opgepakt. De Scrum-master faciliteert en de teamleden bepalen zelf hoe de taken worden verdeeld en wie wat doet. Door de daily stand ups weet iedereen wat iedereen heeft gedaan en van plan is te doen om de afgesproken workload binnen de iteratie af te krijgen. Aan het eind van elke iteratie wordt een demo gegeven van wat er wordt opgeleverd en er wordt een sprint review gehouden om te kijken wat er goed is gegaan en wat er kan worden verbeterd. Daarna start een nieuwe iteratie met de sprint planning waarin wordt bepaald hoeveel workload het team weer aankan.



Dit heb ik overal gezien waar ik in een Scrum-ontwikkelproject heb gewerkt. Wat echter niet anders is dan in een waterval ontwikkelproject, is de manier waarop het werk wordt uitgevoerd. Er wordt nog steeds eerst een ontwerp opgesteld, waarna alle onderdelen die in de iteratie gepland zijn, worden ontwikkeld en als laatste komt testen er achteraan om het hek ➔

af te sluiten. En als er genoeg tijd over is voor test, dan wordt alles uit de huidige iteratie getest opgeleverd. Want de discipline test bepaalt uiteindelijk toch wat er opgeleverd wordt. Dit is echter niet dé gedachte achter Scrum. Deze is dat alle disciplines samenwerken aan een onderdeel binnen de iteratie totdat die af is en dan pas wordt het volgende onderdeel opgepakt. Hoeveel van jullie hebben dit echt ook zo meegemaakt?



Figuur 3. Waterval-aanpak

Wat moet er gebeuren om de omslag te kunnen maken van de waterval aanpak naar de Scrum-aanpak en waarom zou je dat willen? Neem nou mijn team als voorbeeld. Ze hebben allemaal wel van Scrum gehoord en een paar hebben er wel over gelezen, maar lang niet iedereen heeft zich er goed in verdiept of een cursus gevolgd. 'En is dat nou nodig?' vraag je jezelf af. Waarom zou iedereen die in een Scrum-team werkt, een cursus volgen? En is het volgen van een cursus wel genoeg?

Daarbij ga ik er nu even gemakshalve van uit dat goed na is gedacht over de Scrum-aanpak en dat het past bij het project. Een cursus Scrum zou niemand misstaan. Het geeft je niet alleen de basis om ermee aan de slag te gaan, het geeft je ook het inzicht dat nodig is om de principes achter Scrum te snappen en toe te kunnen passen. Ik heb zelf de cursus Scrum Foundation gevolgd. Dat was heel leuk om te doen, omdat de cursus zelf ook helemaal volgens de Scrum-methodiek is gegeven. Een aanrader dus.

Het belangrijkste voordeel van Scrum, vind ik, is dat je in korte tijd het resultaat van je werk in werking ziet. Wat ook mooi aan Scrum is, is dat je vrijwel continu bezig bent om verbeteringen in je werk door te voeren. Wat is er nou beter dan dat?

Daarnaast moet in mijn ervaring de mindset in veel, en misschien wel in alle, Scrum-teams nog veranderen om te kunnen zeggen dat ze helemaal volgens Scrum aan het ontwikkelen zijn. Hoe kun je de mindset, wat ik in het begin met een jasje heb vergeleken, nou veranderen?

Het is moeilijk voor velen om een jasje waaraan ze gehecht zijn geraakt, uit te doen en een ander jasje te proberen, ook al ziet de andere jas er veel warmer en nieuwer uit. Ik moet toegeven dat ik mezelf er ook op betrap. Het is zo eenvoudig om het oude jasje aan te doen, je weet hoe het zit en wat je eraan hebt. Een ander jasje aandoen is toch wel wennen, maar is het niet precies ook zo gegaan toen je het oude jasje voor het eerst kreeg? Hoeveel moeite kost het je dan om het andere jasje een kans te geven? Weinig. Het is gewoon een kwestie van doen, de voordelen ervan inzien en wennen aan dit nieuwe jasje. Op een gegeven moment ben je er zo aan gewend dat je niet meer beter weet. Willen is kunnen en kunnen is (nog meer) willen! Wel… totdat de volgende methodiek zich aandient, want er zullen altijd wel nieuwe jasjes in de mode komen. ←

SIMPEL AAN DE SLAG MET REGRESSIETESTEN

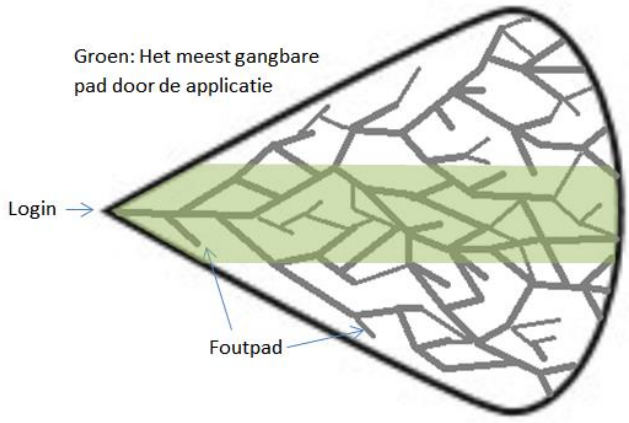
Door Sander Mol ● sandermol@chello.nl

Twee jaar geleden trad ik in dienst bij een nieuwe werkgever. Mijn opdracht was om het gestructureerd testen een plek te geven in de organisatie. Inmiddels heb ik met hulp van collega's een aanpak opgesteld, waarmee de basis van het testvak wordt toegepast op zowel grote als kleine wijzigingen. Bij ons bedrijf werkt dit erg goed.

Daarom heb ik in juni 2013 deze aanpak vertaald naar een nieuwe website, www.simpeltesten.nu. Door deze standaard aanpak aan te bieden, probeer ik iedereen die geïnteresseerd is op weg te helpen met een simpele testaanpak. Geen voorkennis nodig, alleen interesse in kwaliteit. En dan niet alleen de kwaliteit van wijzigingen, maar ook van bestaande functionaliteit.

Regressietesten

Zeker als je tot de lijnorganisatie behoort, is regressietesten een extra belangrijk onderdeel van het werk. Eén deel van een dergelijke test richt zich op de functionaliteit direct om de wijziging heen. Dit artikel gaat echter over het andere deel; het regressietesten van de gehele applicatie. Al zul je natuurlijk niet álles uitgebreid testen; je richt je vooral op de belangrijkste functionaliteit. Om dit wat duidelijker te maken, hanteer ik in mijn trainingen het volgende plaatje.



Voor het gemak begint het gebruik van de applicatie op één plek; bij het inloggen. Daarna gaan de gebruikers allerlei kanten op. Toch maken ze voor 95% gebruik van de rechttoe, rechtaan functionaliteit, weergegeven in de groene zone. Het is logisch dat dit goed moet werken om de bedrijfsvoering te ondersteunen. Veel testen dus. Maar het is óók van belang om de witte zone te testen. Hier zitten de minder gebruikte functionaliteiten. Deze dien je minder te testen, maar zeker niet over te slaan. Je wilt voorkomen dat hier de applicatie triviale foute bevat, vastloopt of bewust verkeerd gebruikt wordt. Én het geeft inzicht in hoe robuust de software gebouwd is. Allemaal nog eens extra belangrijk als je een complexe applicatie wilt testen met een keten van meerdere applicaties of modules.

Hoe weet je wat je allemaal kunt testen

Er zijn veel manieren om naar kwaliteit te kijken. Naast functionaliteit zijn er veel non-functionals, waar je expliciet een oordeel over wilt geven. Met hulp van een simpele matrix delen we alle applicaties op in modules. Daarna ➔

kijken we bij iedere module welke vormen van kwaliteit belangrijk zijn en hoe groot de kans is dat we die kwaliteit niet gaan krijgen.

		Functionaliteit	Gebruiksvriendelijkheid	Inpasbaarheid	Performance	Beveiliging	Continuïteit	Beheerbaarheid
Artikelen module	Belang (impact)	+++	+++	+	++	++	++	+
	Faalkans	+	++	+	++	++	+	+
Winkelwagen module	Belang (impact)	+++	+++	+	++	++	++	+++
	Faalkans	+	++	+	++	++	+	+
Bestel module	Belang (impact)	+++	+++	+	++	+++	++	++
	Faalkans	+++	++	+	+	++	+	+
Financiële module	Belang (impact)	++	+	++	+	+++	++	++
	Faalkans	++	++	++	++	++	++	++
Logistieke module	Belang (impact)	++	++	+++	++	+++	++	+++
	Faalkans	+++	++	++	+++	+++	+++	+++
IT beheermodule	Belang (impact)	++	+	+	++	+++	++	++
	Faalkans	++	++	+	++	++	++	++

De volgende stap is een directe vertaling naar een teststrategie. Ofwel, hoe gaan we de grootste regressierisico's wegnemen door documenten te reviewen, door een systeemtest uit te voeren en door acceptatietesters te laten testen. De donkerblauwe cellen geven een hoog risico aan en zullen dus zwaarder gereviewd en getest worden. Toch is het geen wiskundige vertaling; het blijft belangrijk om zelf de testzwaarte te bepalen. ➔

		Functionaliteit	Gebruiksvriendelijkheid	Inpasbaarheid	Performance	Beveiliging	Continuïteit	Beheerbaarheid
Artikelen module	Review	+	++		+	+		
	Systeemtest	++	++		++	++	+	
	Acceptatietest	+	+++					
Winkelwagen module	Review	+	++		+	+		++
	Systeemtest	++	++		++	++	+	
	Acceptatietest	+	+++					++
Bestel module	Review	+++	++			++		
	Systeemtest	+++	++		++	+++	+	
	Acceptatietest	+	+++					+
Financiële module	Review	++		++		++	+	++
	Systeemtest	++			+	+++	++	
	Acceptatietest	+	+	++				++
Logistieke module	Review	++	+	+++	+++	+++	+++	+++
	Systeemtest	+++	+		+++	+++	+++	++
	Acceptatietest	++	++	+++	+			+++
IT beheermodule	Review	+			+	++	+	++
	Systeemtest	++			++	+++	++	
	Acceptatietest	++	+					++

Vertaling naar een 'testcharter'

Een dergelijke matrix geeft een mooi overzicht van wat er belangrijk is, maar alleen met 'plusjes' kun je nog niets testen. De volgende stap is dus om een lijst met testactiviteiten op te stellen. Dat zou er voor de acceptatietesters als volgt uit kunnen zien.

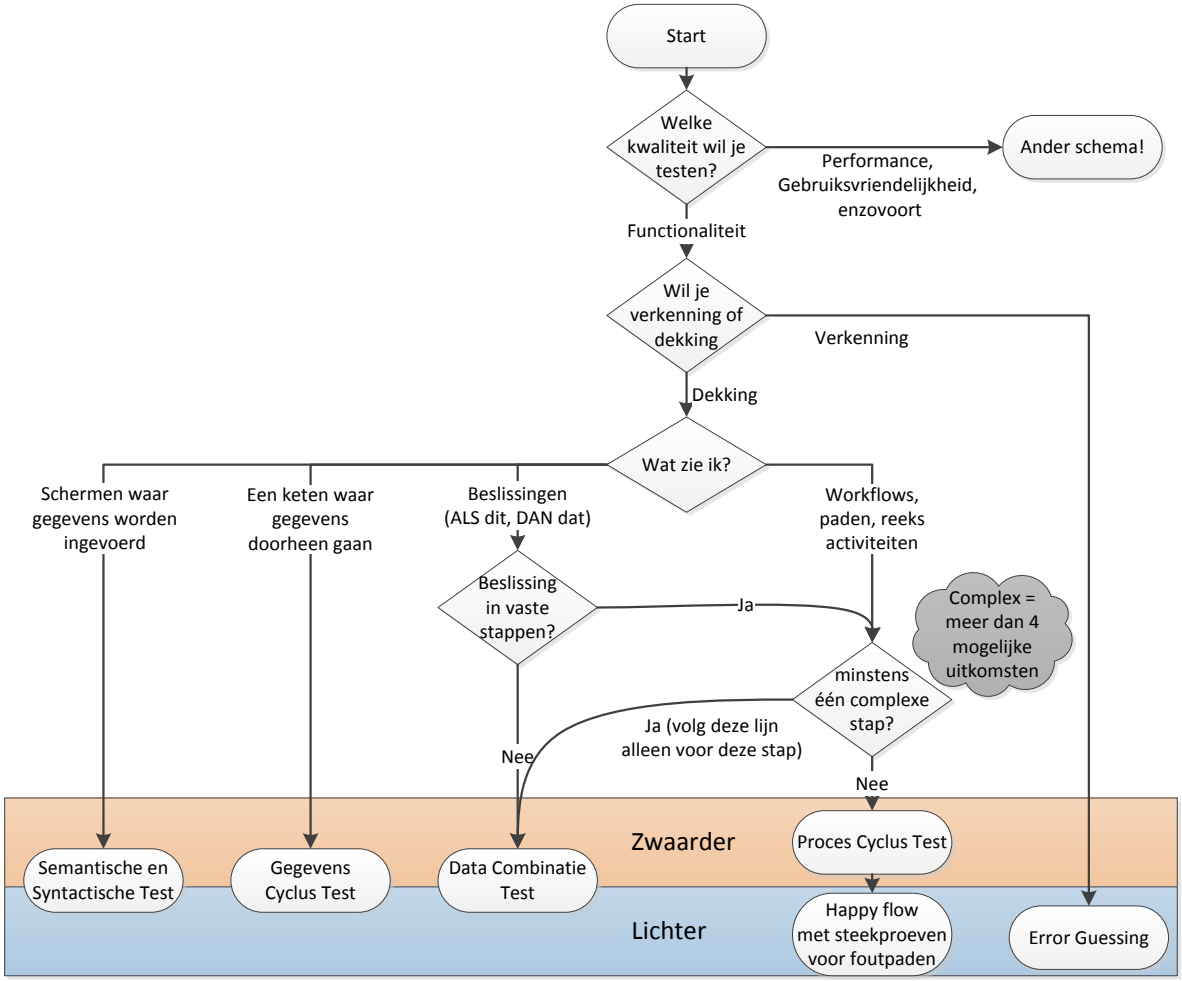
Acceptatietest				
Dag	**Test**	**Activiteit**	**Doorlooptijd**	**OK?**
1	Error guessing	Op basis van ervaring navigeert de verkoopafdeling door diverse schermen voor een eerste indruk van de gebruiksvriendelijkheid van de modules waar de klanten ook komen. De logistieke afdeling navigeert alvast even door de logistieke module voor een eerste indruk. Beide gebruikersgroepen letten meteen ook alvast op performance.	60 minuten	
1	Functionele controle	De verkoopafdeling test de verkoop van één product voor een bestaande klant en één voor een nieuwe klant, om vertrouwen te krijgen in de functionaliteit. De logistieke afdeling controleert vervolgens de logistieke afhandeling.	30 minuten	
1	Scherm layout	Alle klantmodules worden uitgebreid doorgelopen; op ieder veld worden de verwachte waarden en de uitgesloten waarden gecontroleerd.	180 minuten	
1	Aansluiting bij klanten	Uitnodigen van klantpanel; navigatie analyseren en opmerkingen verzamelen	180 minuten	
2	Aansluiting bij logistiek	Alle key users van Logistiek lopen gezamenlijk het hele logistieke proces door, met de nadruk op producten en bestelopties waar het meeste is gewijzigd.	90 minuten	
2	Beheer batches	Logistiek medewerker draait en controleert de verzending- en retourbatch	120 minuten	
2	Error guessing	Op basis van alles wat eerder getest is, een extra controle op de onderdelen waar de minste zekerheid is gekregen	60 minuten	

Je ziet de aansluiting tussen de teststrategie en de testactiviteiten. Zo hadden we een aparte regel kúnnen opnemen om gebruikers expliciet de performance te laten testen. Volgens de teststrategie is dit echter niet nodig tijdens de acceptatietest. ➔

Ontwerptechnieken

In een paar pagina's hebben we een mooie vertaling van risico's naar testactiviteiten gemaakt. In de praktijk heb je zoiets in een uurtje opgesteld, waarmee het ook toepasbaar wordt voor kleinere wijzigingen. De testactiviteiten kunnen worden uitgevoerd door de testers, vooral de eerste keer zal dat naar eigen inzicht zijn.

De volgende stap is dan om van eigen inzicht naar een goede dekking en verslaglegging te gaan. We hebben daarvoor een klein overzicht opgesteld. Het laat zien welke ontwerptechnieken we hebben geselecteerd en wanneer die toegepast kunnen worden. Hieronder staat dus een selectie voor de organisatie waar ik werk.



Om deze toepassing van ontwerptechnieken te laten slagen, is het belangrijk om die technieken héél toegankelijk aan te bieden en de eerste paar keer samen met de tester toe te passen. Op die manier stellen je collega's toch een dekkende testset op, ondanks dat ze het waarschijnlijk nooit als hun hobby zullen zien.

Hopelijk hebben deze pagina's vol plaatjes een eerste indruk gegeven van het opzetten van een simpele regressietest. Het geeft in ieder geval een snelle start, waar later altijd nog een verzwaring aan kan worden toegevoegd.

Succes met simpel testen! ←

FAST DELIVERY ON A SLOW TRAIN

Door Marc van 't Veer ● marc.vantveer@polteq.com

De metafoor

'Hoe past het snelle metrosysteem op het traject van een langzame goederentrein?'. Dat is een metafoor voor het combineren van verschillende ontwikkelmethodieken binnen één project met één releasedatum. Daarbij gaat het niet om een voorkeur voor een bepaalde ontwikkelmethodiek, maar om een beschrijving van een situatie die steeds vaker zal gaan voorkomen. Welke moeilijkheden kom je tegen en welke oplossingen kun je toepassen, zodat je als tester je werk goed kunt doen?

De dienstregeling

De dienstregeling van een goederentrein, van emplacement tot eindstation, met alle tussenstations, wordt lang van tevoren vastgelegd. En als je mee wilt, zul je je daaraan moeten houden. De metro daarentegen neem je naar behoefte, bijvoorbeeld omdat het regent. Hoe stem je dan de dienstregeling van de metro af op die van de goederenlijn tussen Rotterdam en het Duitse achterland? Een metro vertrekt bijvoorbeeld elke tien minuten en een specifieke goederentrein eens per dag. Een metroreis vraagt qua timing veel minder planning. Als niet iedereen in de metro past, dan kunnen ze met de volgende mee, met een maximale wachttijd van die tien minuten. Als echter jouw containers niet met de goederentrein meekunnen, dan heb je een serieus probleem, want dan komen ze minstens een dag te laat aan. Die goederentrein moet dus strak worden gepland.

Maar wat gebeurt er als een machinist de metro mist of in een verkeerde wagon stapt, of een wissel stuk is? Dan kan, net als bij de Nederlandse Spoorwegen op een winterse dag, de hele dienstregeling vastlopen. Ook bij de metro moet je het een en ander plannen.

Figuur 1. Metro op de Betuwelijn?

Het project

Het totale project waar ik het over wil hebben, werd uitgevoerd door verschillende leveranciers. Elke leverancier volgde zijn eigen methodiek, zijn eigen 'spoor'. Met meer dan honderd medewerkers werd een nieuw programma live gezet. De backend-systemen zijn in waterval ontwikkeld, de (mobile) website in iteraties en de mobile app Agile. Elke leverancier was verantwoordelijk voor het uitvoeren van systeemtesten en het leveren van support tijdens ➔

de integratiefase. In totaal bestaat de end-to-end-keten uit meer dan twintig applicaties die onderling communiceren via een enterprise service bus. Het totale programma had één centraal go-live-moment.

De uitdaging

Voorwaarde om binnen het project te kunnen overleven is basiskennis van alle drie ontwikkelmethodieken. Maar het is best een uitdaging om je aandacht te verdelen over niet alleen de lange termijn (de waterval) en de korte termijn (Agile), maar ook nog over de losse iteraties op zich. Je moet aan de ene kant dagelijks succesvol weten te testen binnen het Agile project, maar ook nog genoeg tijd overhouden voor de voorbereiding van de watervaltesten. Naast het verschil in planningshorizon verschillen de methodieken ook in de Definition-of-Done. Wanneer ben je bijvoorbeeld klaar binnen een watervalmethodiek (testcases staat op 'passed'?), binnen een iteratieve aanpak (alle iteraties geïntegreerd getest?) en binnen Agile methodiek (de user story werkt?).

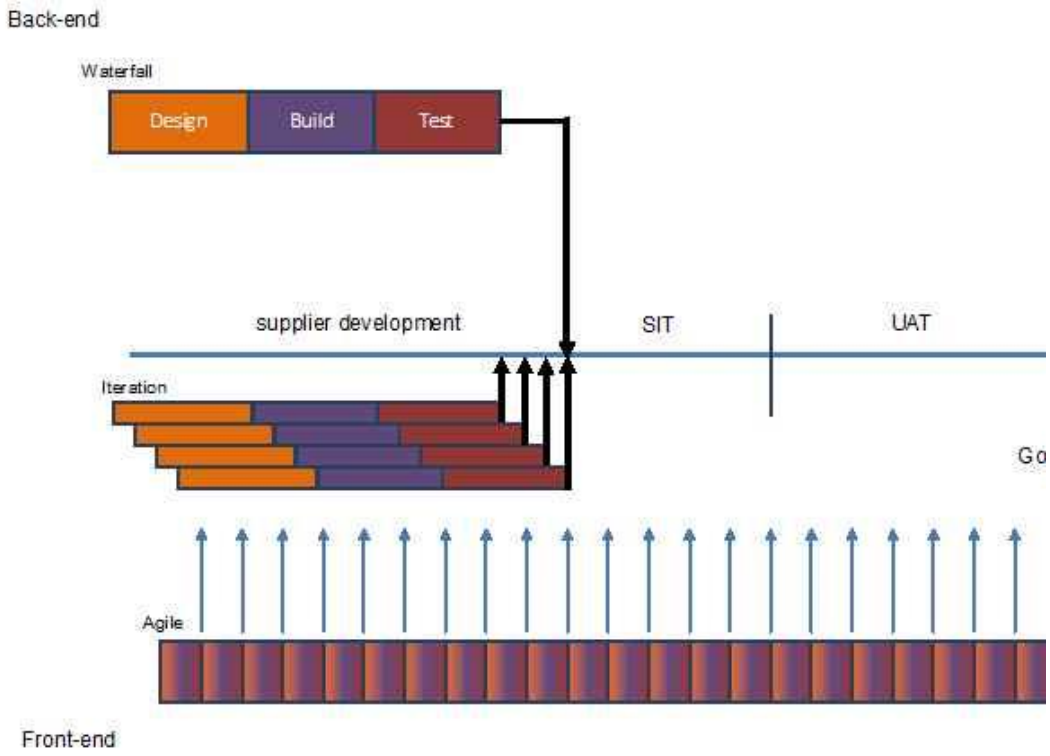
Een tester moet dagelijks schakelen tussen parallel lopende releases en bepalen welke strategie wordt gevolgd om kwaliteit te leveren. Wanneer moet je overstappen, hoeveel overstaptijd of wachttijd is er, of moet je iedere keer een sprintje trekken om soepel te kunnen doorreizen? Het totale dienstrooster en de frequentie van rijden geven het tempo aan, waarin gewerkt moet worden door een tester. Het dienstrooster is de releasekalender. Deze kalender moet kloppen als je hart (heartbeat), zodat kan worden bepaald wanneer je welke trein moet nemen, wanneer je kunt gaan zitten, moet gaan staan of beter kunt blijven staan. Welke taken moet je uitvoeren voor welke release, wanneer moet je voorbereiding klaar zijn, hoe moet je een taak uitvoeren en wie voert een testcase uit op welke omgeving? De uitdaging zit in het flexibel schakelen tussen de verschillende leveranciers, methoden, releases, deadlines, taken en begrippen.

Problemen, risico's en wat het zo complex maakt voor een tester

Een tester is van nature sceptisch, (positief!) kritisch en gezond wantrouwend van karakter. Het eerste wat ik deed, was vragen stellen om de organisatie en het gehele systeem te leren kennen. Voorbeelden: Wat gebeurt er als een defect wordt gevonden, wat als er een verandering in de markt is, een extra change nodig is, een afhankelijkheid tussen systemen is gemist, een deadline wijzigt omdat deze niet wordt gehaald of als er een hele release extra nodig is? Door deze vragen wordt meer duidelijk over hoe de raderen draaien en waar het gaat piepen of kraken.

Problemen

De organisatie waarvoor we dit project uitvoerden, heeft een 'natuurlijke' focus op de logistieke keten en de processen daaromheen. De IT-processen worden bestuurd door middel van een watervalmethodiek. Kijk je als tester naar hun markt, dan zie je veel competitie, stellen de klanten hoge eisen en is 'time to market' veel belangrijker dan het hebben van alle features. Als tester was ik verantwoordelijk voor het coördineren van de gebruikersacceptatietest (UAT), in dit geval zowel voor de business (intern) als voor de eindgebruikers in de markt (extern). De eisen van de eindgebruikers spelen een cruciale rol, wil je als totale organisatie succesvol zijn. Het voelt als tester alsof je een strijd moet leveren om de eisen van de eindgebruiker gerealiseerd te krijgen. Naast de strijd tussen de eigen organisatie en de markt, bestaat er ook een gevecht tussen de verschillende leveranciers. Wiens visie sluit het best aan op de organisatie en de markt, welke oplossing wil iedere leverancier het liefst geïmplementeerd hebben, welke contracten spelen een rol, en welke politieke krachten worden er uitgeoefend? Deze strijd heeft veel invloed op hoe open, soepel en meewerkend alle partijen naar elkaar zijn. ➔



Figuur 2. Parallel ontwikkelen zonder afstemming

Risico's

Niet alleen een tester moet continu zijn aandacht verdelen; dat geldt ook voor een product owner. Een product owner kan niet deelnemen aan alle parallel lopende teams tegelijk. Daarom was in dit geval de product owner niet één persoon, maar een groep die de regie voert over de gestelde eisen. Doordat verschillende onderdelen van het totale systeem door verschillende leveranciers werden ontwikkeld, werden besluiten over de architectuur, interfaces en business rules los van elkaar genomen. Die moeten dan natuurlijk wel onderling worden afgestemd. Het risico is dan levensgroot dat niet alle specificaties van en naar alle leveranciers met elkaar in sync zijn.

Naast de eisen van de business komt de uiteindelijke eindgebruiker in elke methodiek op een ander moment binnen en speelt een andere rol. Binnen de watervalmethodiek is eindgebruiker het eindstation van de ontwikkelcyclus, binnen de andere methodieken wordt de eindgebruiker veel meer en veel vroeger betrokken; hier is de eindgebruiker echt een klankbord. De eisen van de eindgebruiker kunnen de eigen specificaties in een ander daglicht stellen, prioriteiten verschuiven en nieuwe specificaties noodzakelijk maken.

Complexiteit voor een tester

De volgende punten maken het complex(er) voor een tester om in deze situatie te werken:

- Continue wisselen tussen proces en product view;
- Kortetermijn- versus langetermijnfocus;
- Eén wijziging in de planning kan een lawine aan andere wijzigingen veroorzaken;
- Meer machinisten op de trein door een matrixorganisatie;
- Complex communicatieproces;
- Zeer frequente (wekelijkse) software merges;
- Parallel ontwikkelde releases; ➔

- Vaak heel hard rennen voor meerdere teams en dan een dag stil staan;
- Continu scope-discussies en verschuiven van verantwoordelijkheden.

Aan het einde van de dag, week, maand, als alles weer is afgestemd, vraag je je als tester af: is er nog tijd genoeg om alle tests uit te voeren en hoe krijg ik dat gepland? Als tester signaleer je defects en heb je veel invloed op de wijze waarop leveranciers samenwerken. Je bent de boodschapper van het resultaat van veel hard werk. Op welke manier kun je bijdragen aan het soepeler afstemmen van de verschillende manieren van werken?

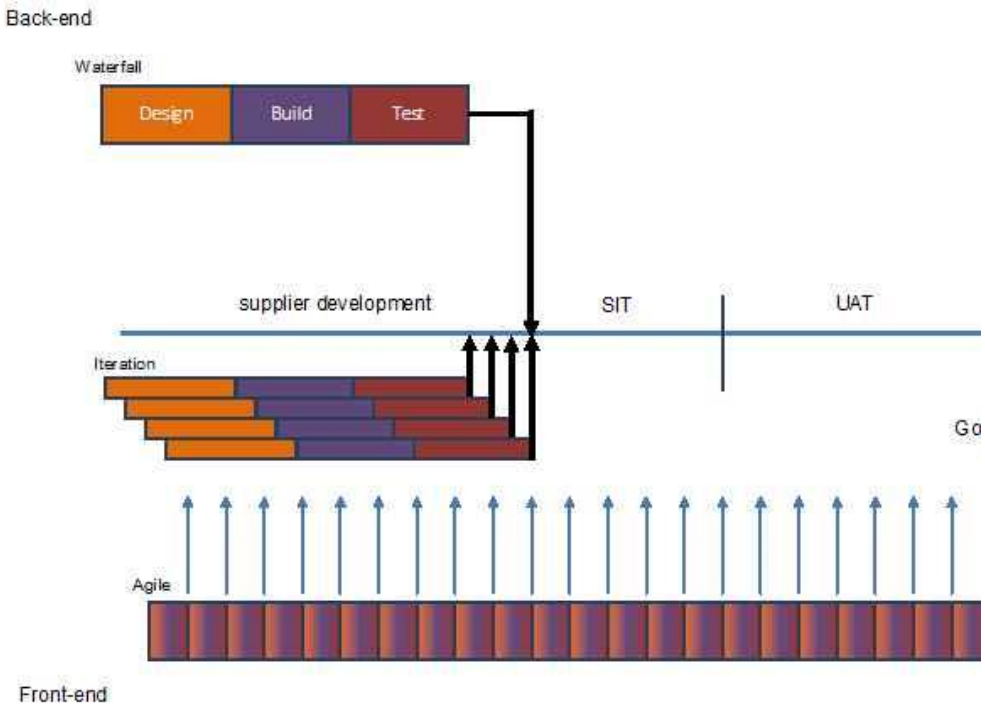
Oplossingen

Als tester wil je samenwerken met en service leveren aan alle leveranciers. Ik geloof dat alleen door samen te werken kwaliteit kan ontstaan. Daarnaast geloof ik er niet in dat één ontwikkelmethode de 'beste' is voor alle leveranciers. Het doel is niet om het perfecte proces te volgen, maar om gezamenlijk een kwaliteitsproduct op te leveren. De oplossingen, die je als tester helpen om te kunnen werken in zo'n complexe situatie, zijn volgens mij te vinden in de flexibiliteit van schakelen in je hoofd, in de methoden die je toepast, in de omgevingen en de tooling die worden gebruikt. Daarnaast kun je bepaalde zaken zo organiseren dat ze de dagelijkse praktijk van een tester helpen (enablers). Ook kunnen aanpassingen in de drie ontwikkelmethoden van pas komen.

Enablers

Voorbeelden van enablers die voor mij belangrijk zijn gebleken, zijn:

- Defects centraal gemanaged (applicatieproces);
- Een team van product owners voor alle partijen en applicaties;
- Meerdere volledig ingerichte en onderhouden testomgevingen;
- Een standaard raamwerk voor alle interfaces voor alle applicaties;
- Een centraal register van alle interfaces;
- Een centrale hartslag in de vorm van een releasekalender;
- Een centraal discussieplatform voor planning, issues, vragen en designkeuzes;
- Een overzicht van afhankelijkheden (planning en applicaties);
- Een geautomatiseerde regressietestset (inclusief de benodigde tools);
- Overzicht van de voortgang binnen de teams;
- Afstemmingsproces tussen alle teams. ➔



Figuur 3. Centrale releasekalender

Afstemming methodieken

Behalve dat het project voor afstemming kan zorgen, kan dit ook plaatsvinden binnen elke ontwikkelmethodiek zodat deze beter aansluit op de rest. Ik hanteerde verschillende opties om waterval, iteratief en Agile methoden samen te voegen:

- Dezelfde tester zit in het Agile-team en doorloopt vervolgens alle volgende fasen in de waterval;
- Testers in koppels, allebei in een Agile- en in een watervalteam, en onderling wordt het werk verdeeld (input en feedback naar beide teams);
- Meerdere Agile-rondes, daarna een integratiefase en overdracht naar waterval;
- Kleine scope in Agile en grotere scope (ketenintegratie) in waterval;
- Vroege betrokkenheid in het watervalproject om voorbereidend werk te doen en ruimte te maken voor deelname aan het Agile team, daarna weer doorgaan in de watervalfasen.

Ook een tester moet zich wapenen voor een strijd binnen het project. Kijk daarbij naar je kennis (gears in je hoofd) en naar zijn je kunde (gears in je methoden).

Gears

Hier is een aantal voorbeelden van de vlakken waarop ik als tester moest schakelen om in alle teams mee te komen:

- Kennis nemen van (alle!) gebruikte termen en begrippen;
- Kunnen omgaan met wijzigingen tussen verschillende teams;
- Kunnen wisselen tussen een proces- en productfocus;
- Verschillende rollen, taken en verantwoordelijkheden hebben;
- Tegelijkertijd een Scrum-master en meerdere testmanagers hebben;
- Kunnen reizen op de releasekalender. ➔

Het belangrijkste is dat ik mijn werk tegelijkertijd zowel traditioneel gefaseerd kan uitvoeren als deelnemend aan een Agile-team. Het is jongleren in taken die gelijktijdig moeten worden uitgevoerd en het vraagt om veel flexibiliteit. Dat je moet omschakelen klinkt logisch, maar soms merk je niet eens dat je omgeschakeld bent. Je loopt bijvoorbeeld het Agile team binnen en je leest welke 'Conditions of satisfaction' er gelden voor een user story in een sprint en welke taken er nog moeten worden uitgevoerd om als team klaar te zijn. De voortgang is zichtbaar als je in het team zit. De volgende dag ben je een detailtestplan aan het reviewen en testscripts aan het opstellen voor een komende release en ga je morgen de deelacceptatie begeleiden van een onderdeel van de website.

Voorwaarde is dat je veel vrijheid hebt om je eigen rol in te vullen. Je kunt niet full time in een Agile-team zitten en ook niet een volledige applicatie testen. Je rol is breed en dus zijn je taken ook erg breed. Je maakt presentaties, automatische regressietestcases, je begeleidt eindgebruikers, je reviewt user stories maar ook testcases en testplannen, je werkt zelfstandig in een team voor een testcoördinator, soms ben je simpelweg handjes om iets uit te voeren en soms moet je zelfstandig een oplossing uitwerken en invoeren.

Soms merk je dat je nog niet omgeschakeld bent, bijvoorbeeld wanneer je vraagt naar een testrapport over de uitgevoerde testcases in een Agile-team. Grote kans dat je wordt doorverwezen naar het Scrumboard waar de 'gescoorde' user stories hangen. Het moeilijkst is het omgaan met prioriteit van taken van verschillende testmanagers en Scrum-masters. Wat gaat er vóór: de testuitvoer in een Agile-team of de voorbereiding voor de volgende iteraties over twee weken? Mijn oplossing is dan vaak dat ik de testcases die ik uitvoer voor het Agile-team hergebruik voor de andere teams, dus de uitvoer in het ene team is de voorbereiding voor het andere.

Conclusie

Het ideaal is uiteraard dat alle leveranciers perfect samenwerken en een realistische planning afgeven die ook echt wordt gevolgd. Realiteit is dat het een complexe omgeving is om in te werken. Een tester kan de smeerolie zijn om de totale machine soepel te laten draaien. Op het moment dat enablers zoals een centrale releasekalender worden gevolgd en de teams kleine aanpassingen doen aan hun standaard manier van ontwikkelen, dan kan er pas echt kwaliteit worden geleverd. Van de tester wordt daarbij veel gevraagd. Met name moet je heel flexibel en heel breed zijn in de kennis en kunde die je toepast, want alleen zo kun je een optimale bijdrage leveren aan alle teams en aan het totale project. ←

SCHOMMELEN TUSSEN VRIJHEID EN UNIFORMITEIT. SCRUM EN TESTEN BINNEN DE BANK

Door Yvonne van Zaanen ● yvzaanen@zonnet.nl

Na het introduceren van de Scrum-werkwijze bij de Triodos Bank verspreiden de testers zich over de teams. Binnen hun eigen Scrum-team krijgen ze veel vrijheid. Ze kunnen zelf bepalen hoe ze het meest efficiënt en handig werken. Ze zetten tooltjes in, minimaliseren documentatie en beperken waste zoveel mogelijk. De testers zelf zijn erg enthousiast over deze manier van werken. Maar door deze vrijheid gaan het testproces en de ingezette tooltjes steeds meer verschillen per team, met alle gevolgen van dien. Door onder andere de inzet van meer Scrum-teams en (de daarbij behorende) testers, maar ook de toenemende druk van externe toezichthouders ontstaat er meer behoefte aan uniformiteit.

Maar hoe kan die uniformiteit vorm krijgen, als de individuele testers erg blij zijn met hun vrijheid? Hoe ver ga je? Wat en hoe maak je uniform? Welke afwegingen worden daarbij gemaakt?

Overgang

In 2010 is de ICT-projectenafdeling overgegaan van een waterval- naar een Scrum-werkwijze. Er zijn Scrum-teams samengesteld die bestaan uit drie tot vier ontwikkelaars, een informatieanalist en een tester. Het zijn zelfsturende teams en naast de standaard Scrum-inrichting (stand-up, retrospectives, pokersessies, sprints, Scrum-board, etcetera) kunnen ze voor een groot deel zelf richting geven aan het wat, maar vooral het hoe. Een van de Scrum-teamleden fungeert als Scrum-master.

Scrum is geen methodiek die bepaalt welke tools worden gebruikt en schrijft geen templates en documentatie voor betreffende het testproces. Het team is verantwoordelijk voor het eindproduct, en wie wat daarbinnen doet wordt onderling bepaald. De testers zijn testmanager, -coördinator en tester binnen het Scrum-team. Als ervaren testers hebben ze ieder hun eigen ideeën en inspiratie over wat het meest efficiënt en handig werkt. Ze fungeren vol overgave als ‘testgeweten’ van het team. ➔

Effecten

Uiteraard heeft de invoering van Scrum een heleboel positieve effecten die specifiek betrekking hebben op het testen en de testdiscipline.

Kwaliteit van het testen is duidelijk verbeterd, testen heeft aandacht vanaf het begin van een project, testers schakelen veel directer met de andere disciplines. Door de iteratieve manier van werken wordt er al zo snel mogelijk aandacht besteed en gedacht aan testen, zoals bij unit-tests; geautomatiseerd (regressie)testen heeft veel aandacht.

Vanuit de business is men meer gericht op samenwerking bij het testen, samen testen met de testers uit het Scrum-team, vanuit het Scrum-team bieden alle disciplines ondersteuning bij de acceptatietests.

Omdat de testers de vrijheid hebben om een betere, meer efficiënte werkwijze te bedenken en direct toe te passen is de ontwikkeling van het testdomein ook volop in beweging. Ook al zet de tester zich nog steeds in voor het 'goed' testen van de opgeleverde software, toch heeft de invoering van Scrum ook wat minder positieve effecten. Om er een aantal te noemen:

- Voor de business is het niet altijd duidelijk hoe het testproces verloopt en wat daarbij van de testers wordt verwacht. Dit verschilt per project en per team.
- De (interne) auditors kijken kritisch naar het testproces als zij input nodig hebben om de kwaliteit van de opgeleverde software te beoordelen. Zij willen meer (vastgelegde) informatie en vertrouwen niet op de 'blauwe ogen van de tester'.
- Nieuwe medewerkers en nieuwe testers krijgen weinig handvatten.
- Het management krijgt niet veel zicht op het testproces, ingebed in de Scrum-manier van werken. Terwijl de tests en uitkomst daarvan hen juist inzicht moeten geven in de kwaliteit van de software en deze een belangrijke rol spelen in de afweging over het al dan niet in productie willen nemen van de software.
- Er zijn geen checklist of iets dergelijks. Testers kunnen soms zaken over het hoofd zien (Load, Stress, Performance, Security), omdat de tester midden in het team staat en niet 'erboven' met een algemene kwaliteitsblik. Het is onduidelijk of en door wie het non-functional testen moet worden opgepakt, dat is niet specifiek belegd.

Van bovenstaande punten zal ik er twee nader toelichten.

- Nieuwe medewerkers
  De laatste twee jaar is het aantal Scrum-teams uitgebreid van vier naar zeven. Er zijn dus ook meer testers in dienst gekomen. In de eerste weken vragen die nieuwe testers naar templates, bevindingenprocedures, enzovoorts. Uiteraard is er wel een teststrategie en een testpolicy. Die zijn echter zo algemeen dat ze daar in de dagelijkse praktijk niet mee uit de voeten kunnen. Na enige gewenning waarderen de testers de vrijheid wel meer, maar de initiële vragen zetten ook de andere testers (die de betreffende persoon inwerken) aan het denken. Er ontstaat behoefte aan meer eenduidigheid.
- Audit
  Tijdens een audit op een van de projecten komt nog eens expliciet aan het licht dat de auditors erg kritisch kijken naar het testproces. Daarbij leggen zij de nadruk op traceability en op 'bewijs'. Om de kwaliteit van het opgeleverde product te meten kijken ze ook naar de resultaten van de tests. Dit kan gaan wringen als binnen
  ➔

- het Scrum-team weinig wordt vastgelegd en er geen standaard testproces is. Uiteraard is het opleveren van goed werkende software erg belangrijk, maar dat kan niet ten koste gaan van documenteren.
- duidelijke verwachtingen naar elkaar zijn belangrijk voor een positief resultaat van een audit. Men moet efficiënt kunnen werken zonder dat het maken van 'meer dan voldoende' documentatie dit proces in de weg zit.

Aanzet

Al met al heerst binnen de testdiscipline het gevoel dat er op verschillende terreinen verbeteringen mogelijk zijn. Dit ook naar aanleiding van een 'testworkshop' die bij de business is gehouden. Aangezien gevoel niet genoeg is om mee aan de slag te kunnen, is er een zogeheten 'lean tea'-sessie georganiseerd. De sessie heeft als doel te kijken welke verbeterpunten op het gebied van testen worden gedeeld door meerdere mensen, deze te prioriteren en er vervolgens mee aan de slag te gaan. De punten zijn eerst door de leden van het testteam op een post-it gezet. Er blijkt een aantal punten te zijn waar meerdere mensen mogelijkheden tot verbetering zien. Die punten worden vervolgens gegroepeerd, de testers kunnen hun voorkeur aangeven voor bepaalde onderwerpen en meeste stemmen gelden. De top vier is vervolgens behandeld.

In willekeurige volgorde:

- De behoefte aan een eenduidig algemeen testproces; ➔

- Uniformiteit en afspraken betreffende de testomgevingen;
- Een visie over de toekomst van de testautomatisering;
- Meer gestructureerde kennisdeling.

Elk onderwerp wordt tien minuten besproken. Aan het eind van de sessie gaat een aantal werkgroepjes met deze topics aan de gang.

Implementeren

Door de sessie op deze manier te organiseren wordt duidelijk dat er vanuit verschillende testers behoefte is aan meer structuur en uniformiteit, overigens zonder de ondertussen geïnstitutionaliseerde vrijheden kwijt te willen. Een toolbox voor testers en testen lijkt een zeer welkome aanvulling voor de testers en handvatten voor de nieuwelingen. Kennisdeling meer structureren en duidelijkheid over testomgevingen en het testproces komt zowel de ervaren tester als de nieuweling ten goede.

Waar wel op gelet moet worden is dat het geen 'testfeestje' wordt. De ICT-projectenafdeling als geheel moet aanhaken, aangezien testen binnen Scrum een verantwoordelijkheid is van iedere discipline en niet alleen van de tester.

Grenzen

Hoe bepaal je wat wel en wat niet uniformiteit behoeft? Enthousiast begonnen met een algemene testaanpak komen we toch wel regelmatig 'indien van toepassing', 'eventueel' en 'afhankelijk van het project' tegen. En of iets van toepassing is, bepaalt de tester en het Scrum-team. Wat in ieder geval wel toepasbaar moet worden gemaakt, is de afstemming tussen de verschillende soorten testen en de traceability van het testproces. Door het opstellen van de testaanpak blijkt dat er een aantal lacunes is. Zo is de overdracht van de testware iets wat helemaal niet meer aan de orde komt in de teams, terwijl daar vanuit de beheerorganisatie wel behoefte aan is. Daarnaast komt naar voren dat er niet eenduidig of helemaal niet (meer) risico gebaseerd wordt getest.

De testers kunnen zich vinden in het testaanpak-document en zien er zeker de voordelen van in. Aangezien het testaanpak-document is opgesteld vanuit de praktijk en door de testers zelf, zijn de tips en trucs, handvatten en andere aanbevelingen een welkome aanvulling om nog beter, bewuster en efficiënter aan de slag te gaan.

Conclusie

Het is goed om regelmatig erbij stil te staan waar ieder mee bezig is en wat er kan worden verbeterd. Verbetering is een continu proces en niet iets wat af en toe moet gebeuren of door incidenten wordt gedreven. Ook in een Agile-Scrum-omgeving is er behoefte aan uniformiteit, niet zozeer op het gebied van templates, maar meer op het gebied van werkwijze en grenzen waarbinnen de vrijheid zich bevindt.

Daar regelmatig bij stilstaan en bijsturen waar mogelijk vergroot het werkplezier zonder dat er al te veel aan de 'vrijheid' hoeft te worden getornd. Als testdiscipline vormen we een Gilde binnen de ICT-organisatie. Het is een opdracht binnen dit Gilde om van elkaar te leren, ontwikkelingen in het testvak te delen en samen (met input van de business en andere disciplines) te bepalen waar we de grenzen leggen.

Uniformiteit en diversiteit kunnen bij een Scrum-werkwijze heel goed hand in hand gaan. ←

TESTVERBETERING DOE JE OP DE WERKVLOER

Door Ben Visser (links) ● ben.visser@sogeti.nl @benvissersogeti

Gerrit de Vries (midden) ● gerrit.de.vries@sogeti.nl

Eveline Moolenaars (rechts) ● eveline.moolenaars-koetsier@sogeti.nl

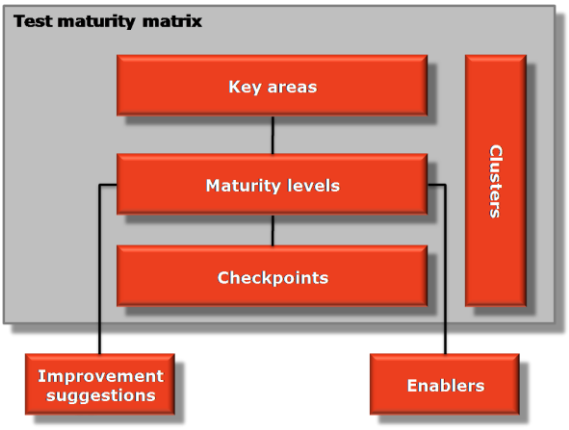
Nederland is een goed ontwikkeld 'testland'. Niet alleen is de gemiddelde tester (voor zover die bestaat ;-) gecertificeerd in TMap®, ISTQB®, Agile, ... etcetera, ook is het besef sterk aanwezig dat de wereld constant verandert en dus ook de 'testwereld'. Net als voor 'gewoon testen' hebben we ook voor het veranderen mooie modellen en bewezen aanpakken (TPI NEXT® , TMMI®, ...). Op het TestNet voorjaarsevenement presenteren wij een overzicht van de trends en ontwikkelingen die we de afgelopen jaren hebben geconstateerd aan de hand van tientallen TPI NEXT assessments bij bedrijven. Maar testverbeteren hoeft niet altijd te beginnen met een expliciet assessment, sterker nog: wij zien vrijwel dagelijks dat testverbeteren een mindset is bij testers, bijna een tweede natuur, die we met ons allen elke dag weer op de werkvloer ten toon spreiden. Het 'ongemerkt' kleine verbeteringen doorvoeren, zonder expliciet verbeterbudget of opdracht van een projectleider of afdelingshoofd, vindt heel vaak plaats. En juist ook voor die tweede natuur is TPI NEXT een nuttig hulpmiddel. Dit artikel gaat in op de manier waarop je TPI NEXT als hulpmiddel kunt gebruiken voor 'Continuous Test Improvement'.

Kort door de bocht kan TPI NEXT op twee manieren helpen:

- Door inhoudelijke hints en tips, op basis van de verschillende elementen van het TPI NEXT model.
- Door trends en ontwikkelingen visueel te representeren, op basis van de Maturity Matrix of van de Benchmark Spider. ➔

TPI NEXT ondersteunt inhoudelijk bij testverbeteren

De kern van TPI NEXT is de maturity matrix die uit 157 controlepunten is opgebouwd. Elk controlepunt op zich is al een impliciete verbeterhint of –tip, maar daarnaast kent TPI NEXT nog twee elementen die buiten de matrix vallen: verbetersuggesties en (bij gebrek aan een adequate Nederlandse term) 'enablers'. Beide elementen kunnen op elk moment in de levensloop van een project of sprint handvatten bieden om geconstateerde knelpunten op te lossen.



Verbetersuggesties

De verbetersuggesties zijn situatiespecifieke beschrijvingen van hoe een verbetering bereikt kan worden. Situatiespecifiek mag je in dit kader lezen als 'Aandachtsgebied en Volwassenheidsniveau' specifiek: een verbetersuggestie heeft betrekking op bijvoorbeeld het doeltreffender of doelmatiger maken van de activiteit rapporteren. Het grote onderscheid tussen controlepunten en verbetersuggesties is dat controlepunten zoveel mogelijk context onafhankelijk zijn geformuleerd (ze richten zich op het 'wat'), maar dat bij verbetersuggesties de context juist bepalend is (ze richten zich op het 'hoe'). Controlepunt 1 van aandachtsgebied Rapportage, volwassenheidsniveau Gecontroleerd geeft aan dat de rapportage zowel aspecten van tijd, kosten, resultaat als risico dient te bevatten. Hóe je dat realiseert, is niet van invloed op het al dan niet aanvinken in de volwassenheidsmatrix . Het boek[1] – en dus de verbetersuggesties die er momenteel in staan – dateren uit 2009, toen Agile nog lang niet zo hot was als nu. Het gebruik van de Definition-of-Done om acceptatiecriteria en (delen van) de teststrategie vast te leggen kan – binnen de context van bijvoorbeeld Scrum - een uitstekende verbetersuggestie zijn. Maar binnen de context van een geografisch verspreide ketentest met een doorlooptijd van misschien wel twee maanden is er van een DoD waarschijnlijk helemaal geen sprake. Dan zul je de acceptatiecriteria en teststrategie op een andere wijze vastleggen.

Enablers

Testen staat niet op zichzelf, het is nauw verbonden met de manier waarop systemen ontwikkeld worden. De relatie van testen met andere IT-disciplines, andere IT-activiteiten en niet-testproducten vatten we samen onder het paraplubegrip enablers. Enablers zijn aandachtsgebied- en volwassenheidsniveau-afhankelijk, net als verbetersuggesties. Een voor de hand liggende enabler voor testplanning is de overkoepelende activiteit 'projectplanning': een projectplanning die gebaseerd is op heldere uitgangspunten en concrete kengetallen biedt een beter startpunt voor een goede testplanning dan een schuivende planning met onduidelijk afhankelijkheden.

Enablers zijn krachtig bij het identificeren van quick wins en werken ook 'de andere kant op'. Door vanuit testen zelf het goede voorbeeld te geven, lukt het beter goede randvoorwaarden te creëren dan met activiteiten die we zelf ook niet onder controle hebben. ➔

---

[1] *TPI NEXT® Business Driven Test Improvement, G. de Vries, B. Visser e.a., 2009*

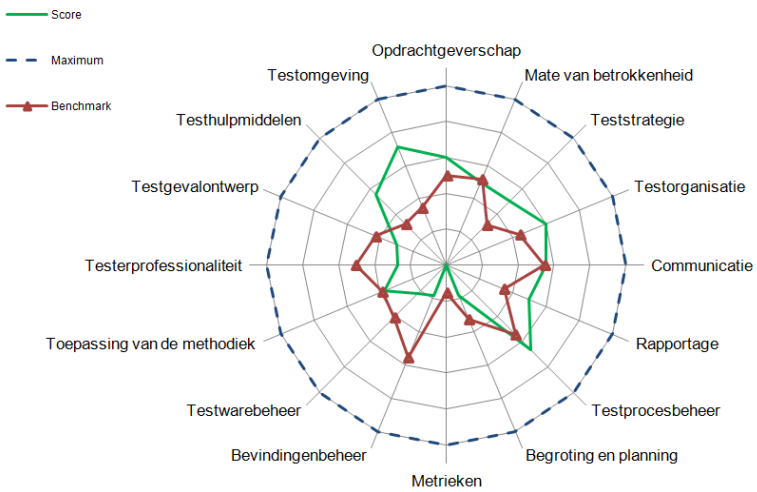
Representeer trends en ontwikkelingen visueel

In de vorige paragraaf hebben we op een kwalitatieve manier aangegeven hoe TPI NEXT je verbeteringen op de werkvloer ondersteunt. Maar het is ook erg prettig om verbeteringen kwantitatief inzichtelijk te kunnen maken. TPI NEXT biedt twee grafieken om het volwassenheidsniveau van testen inzichtelijk te maken: staafdiagrammen en spider-diagrammen. We gebruiken deze grafieken op twee manieren voor 'testverbetering op de werkvloer'.

De eerste manier is het visualiseren van één of meerdere 'self assessments'. Als je al een tijdje binnen een domein test, heb je geen interviews en literatuur onderzoek nodig om de controlepunten van de maturity matrix te kunnen scoren. Het is meestal ook niet nodig om alle controlepunten te scoren, als je na vijf of zes controlepunten constateert dat er maar één of twee op 'ja' staan, dan zijn de volgende controlepunten niet relevant voor eventuele verbeteracties. Die kan je dus (voorlopig) negeren. De pieken en (met name) de dalen in de grafieken zijn interessant, die geven aan waar – volgens jou! – het testen meer of minder volwassen is. En bieden richting om – bijvoorbeeld met behulp van de eerder beschreven kwalitatieve ondersteuning vanuit TPI NEXT – verbeteracties te initiëren.

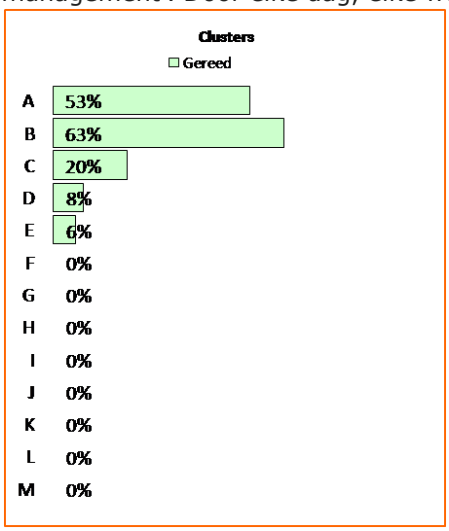
Als je na verloop van tijd – na twee weken, na één maand, niet veel langer: na twee maanden is al lang! - een vervolg self assessment doet, ben je nog veel sneller klaar. De aandachtsgebieden/controlepunten waar je geen verbeteracties voor hebt uitgezet kun je namelijk overslaan. Die zijn in de korte tussenperiode echt nog niet drastisch veranderd.

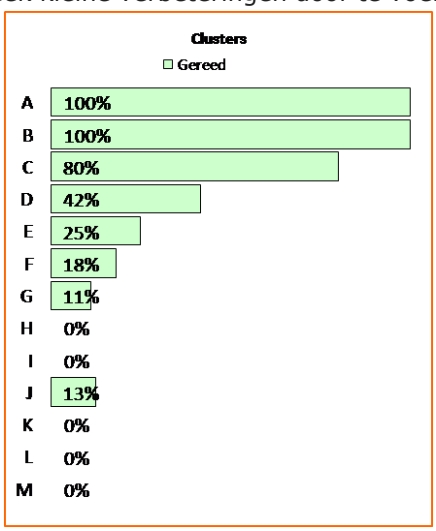
De tweede manier waarop we grafieken gebruiken, is om de volwassenheid van onze testactiviteiten te vergelijken met een industriegemiddelde. Dat gemiddelde kan van 'de hele IT-gemeenschap' zijn, maar ook bijvoorbeeld van alleen de financiële sector of de sector transport. ➔



Houd de resultaten niet voor jezelf

Eén van de grote voordelen van testverbetering op de werkvloer is dat je niet afhankelijk maakt van 'het management'. Door elke dag, elke week kleine verbeteringen door te voeren vergroot je je eigen grip op je werk en (dus!) het plezier in je werk. Maar de effectiviteit en houdbaarheid van 'klein bezig zijn' heeft zijn grenzen. Dus draag je resultaten uit, vier de succesjes die je boekt. Want de grootste vijand bij het verbeteren van 'testen' zijn we misschien zelf wel, kritisch als we zijn op onszelf, altijd manieren vindend om nog beter te testen, maar ondertussen vergetend waardering voor onze toegevoegde waarde op te eisen, juist bij het management! ←





TESTNET NIEUWS

TestNet Nieuws is een uitgave van de Vereniging TestNet, een bloeiende vereniging met meer dan 1600 professionele testers als lid. TestNet streeft de professionalisering na van het testen van IT-producten, en de vergroting van de bewustwording en het belang van testen als vak apart. TestNet stimuleert het uitwisselen en uitdragen van vakkennis, ervaring tussen vakgenoten en stimuleert onderzoek vanuit zowel wetenschappelijk als praktisch perspectief. Voor meer informatie en lidmaatschap, zie http://www.testnet.org.

TestNet Nieuws brengt tweemaal per jaar een Special uit, met artikelen over een actueel thema uit de testwereld, gerelateerd aan het TestNet Voorjaars- en Najaarsevenement.

Daarnaast verschijnt op internet de TestNet Nieuws Weekly, een blog met iedere week een artikel over testen en TestNet. ←