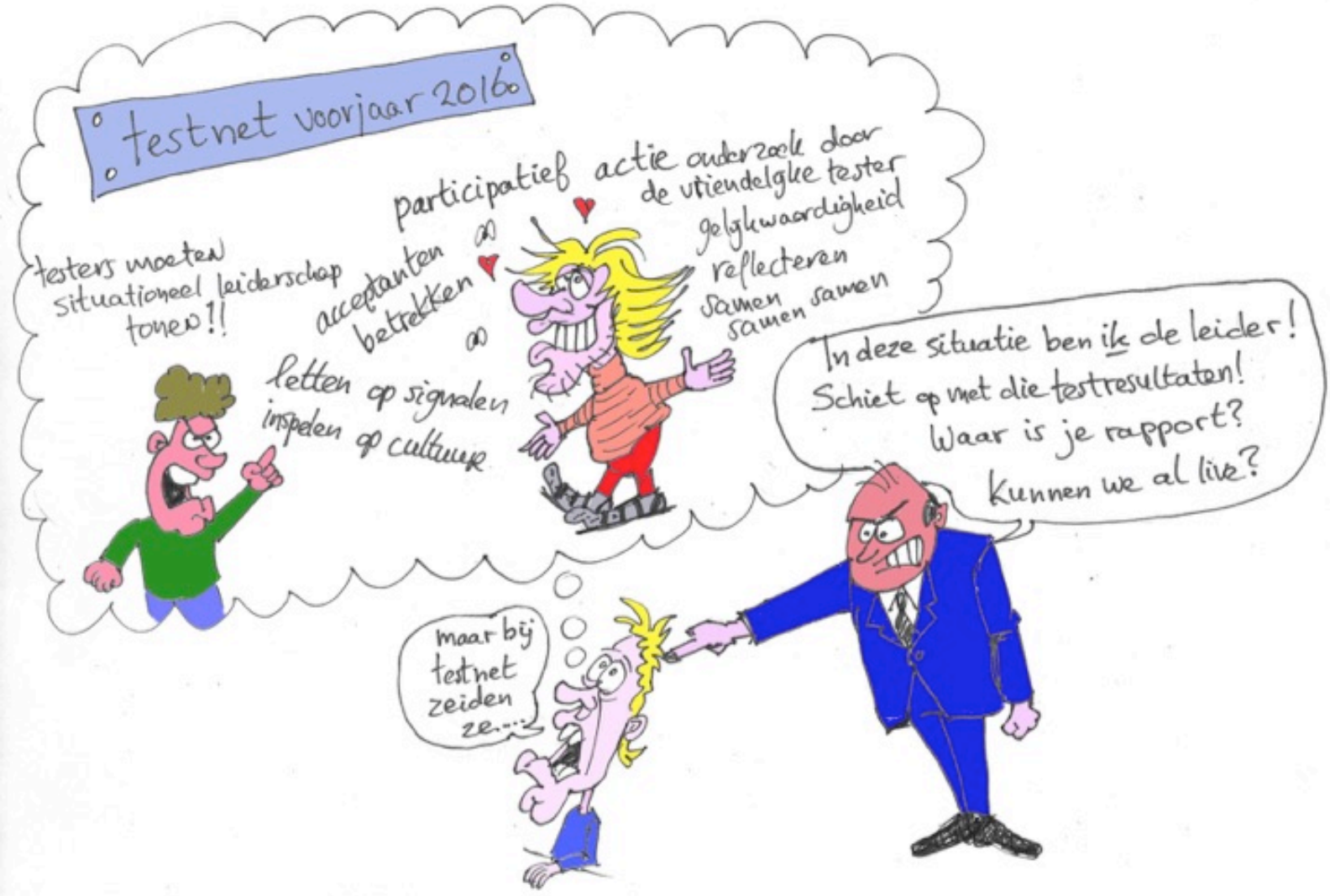
TESTNET NIEUWS

het online en interactieve testmagazine



Voorjaarsspecial 2016
Verbreed je basis: nieuwe vaardigheden voor testers!

TESTNET NIEUWS

Mei 2015 • Jaargang 20 • Voorjaarsspecial

www.testnet.org secretaris@testnet.org

VAN DE REDACTIE

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

Vaak hoor je over nieuwe zaken in een vakgebied als opmerking: 'Oude wijn in nieuwe zakken'. De zaken zijn anders gepresenteerd, maar niet wezenlijk veranderd. Voor veel trends en innovaties in de IT lijkt dit inderdaad zo. 'Agile deden wij twintig jaar geleden al, maar we noemden het niet zo', is een opmerking die nog wel eens valt. Wat is twintig jaar nu eenmaal? Een korte periode waar, als je er relatief op terugkijkt, erg veel veranderd is. Testen is volledig in de belangstelling, in teams werken op een iteratieve manier is bepalend voor een overgroot deel van de IT, het internet als medium is niet meer weg te denken en daardoor is de techniek ook aardig veranderd om software te bouwen en te onderhouden.

Het is goed dat veel zaken al eens eerder zijn uitgevonden en nu 'opnieuw worden uitgevonden', progressie zit niet alleen in nieuwe uitvindingen. Ik denk dat de kracht zit in het verbeteren van bestaande ideeën, zodat we het geleerde uit het verleden kunnen gebruiken. Fouten uit het verleden hoeven niet opnieuw gemaakt te worden. 'Het wiel opnieuw uitvinden' kan goed zijn als je maar goed onderzoekt hoe het wiel in het verleden is uitgevonden. Daar zit een verschil tussen amateuristisch en professioneel werken. De professional kent de geschiedenis van zijn vakgebied en kan daar op voortborduren.

Vroeger had je 'testers'. Tegenwoordig heb je bijvoorbeeld ook security-, usability-, performance- en testautomatiseringspecialisten die het vakgebied verbreden. Niet al deze specialisten komen uit de testwereld, maar komen uit gebieden als marketing, beveiliging of zijn programmeurs.

Interessante gebieden om naar te kijken vanuit het beroep van 'functionele' tester. Wellicht dat deze uitbreidingen op het vakgebied je wel meer liggen dan je nu doet. Het begint bij verbreden, maar wellicht is het nog interessanter om je te *verdiepen* en te *specialiseren*?

In dit magazine een voorproefje op het voorjaarsevenement waar je (alweer!) een overzicht krijgt van de veelzijdigheid van ons geweldige vakgebied. Veel leesplezier en tot ziens op het voorjaarsevenement! ←

COLOFON

Redactie
Gerben de la Rambelje
Gilbert Smulders
Hans Lantink
John Kronenberg
Kees Blokland
Lisa Gelijns
Paul Beving
Rob van Steenbergen
redactie@testnet.org

Bestuur

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

In dit nummer

VAN DE VOORZITTER

Door Rik Marselis ● voorzitter@testnet.org

Ons aller evenementencommissie is in ieder geval tot de conclusie gekomen dat je nieuwe vaardigheden nodig hebt, blijkens het thema van het voorjaarsevenement. En dit thema heeft ook veel schrijvers geïnspireerd zoals je ziet in deze mooie editie van het TestNetNieuws magazine.

Een term die tegenwoordig veel wordt gebruikt is de 'T-shaped' professional, waarover NGI-NGN, Connect.FRL en TestNet samen een thema-avond hebben georganiseerd op 19 april. En daar komen weer varianten op zoals de 'Pi-shaped' (naar het wiskundige teken voor Pi) en 'Comb-shaped' (een liggende streep met heel veel verticale streepjes). In een ochtendworkshop op het evenement hoor je hier vast veel meer over.

En voor je het weet worden er zoveel vaardigheden veronderstelt dat zelfs de supertesters die we een aantal jaar geleden op het TestNet-podium hadden, er niet meer aan kunnen voldoen.

De discussie die op het najaarsevenement van 2015 door Rini van Solingen werd aangewakkerd, namelijk het eind van de functie van tester, zal nog wel even duren. Voor mij staat echter als een paal boven water dat in welke functie of rol dan ook, het VAK 'testen' alleen maar belangrijker wordt. Testen was, is en blijft een heel belangrijk onderdeel van de zorg dat een IT-product de juiste kwaliteit heeft en geen onaanvaardbare risico's met zich brengt, zodat de eindgebruikers met vertrouwen een toegevoegde waarde aan zo'n IT-product zullen ontlenen.

Stilstand is achteruitgang, is een oude wijsheid. Dus alleen al daarom is het goed je te verdiepen in nieuwe vaardigheden zoals die in deze TNN en op het voorjaarsevenement worden geboden. Maar daarnaast is het evenement ook leuk! Je wordt uitgedaagd om zelf allerlei specifieke aspecten van het vak te verdiepen, zoals programmeren in Python, Security Testen, Model-Based Testen, Data Profiling en het testen van RESTful webservices. Daarnaast mag je leuke dingen doen, zoals spelen met Lego (maar wel serious...) en je mag creatief destructief zijn met Scrum.

Ik ben erg blij dat we twee top-testers als keynote-sprekers hebben kunnen krijgen. Kristoffer Nordström uit Zweden en Paco Hope, een Amerikaan uit Engeland. En natuurlijk heeft de evenementencommissie op basis van de call-for-papers weer een hele mooie line-up van nieuwe en bekende sprekers in de parallel-sessies bijeengebracht. Met zelfs sprekers uit Duitsland en Belgie kunnen we met recht spreken van een internationaal evenement. Al kiezen we er als TestNet bewust voor om de voertaal gewoon Nederlands te houden. Want tenslotte zijn we een Nederlandse vereniging. Natuurlijk krijgen we wel eens vragen van buitenlanders die het jammer vinden dat een TestNet evenement voor hen weinig toegevoegde waarde biedt omdat ze de taal niet spreken, maar wij willen ook graag met name 'nieuwe' sprekers op het podium krijgen en dan is presenteren in een andere taal toch weer een extra struikelblok.

Heb jij een goed verhaal over jouw praktijkervaringen en/of visie op het testvak? Stuur dan een voorstel in op de call-for-papers voor het najaarsevenement. Of stuur een idee voor een thema-avond aan de evenementencommissie (evenement@testnet.org). Maar natuurlijk kun je ook in een van de vele werkgroepen meedoen (kijk op de website https://www.testnet.org/testnet/p000014/werkgroepen). Of schuif aan bij een leergang (hierover hoor je binnenkort meer).

Graag wil ik iedereen die heeft meegeschreven aan deze TNN hartelijk bedanken!! En ik wens alle lezers veel inspiratie en plezier.

Afrondend: het testvak is een mooi vak om mee bezig te zijn, je verder in te verdiepen en ervaringen te delen met je vakgenoten. TestNet is daarvoor natuurlijk jouw club!

Veel testplezier! ←

WIL IK MEER VAARDIGHEDEN?

Door Eibert Dijkgraaf ● Eibert.Dijkgraaf@praegus.nl @EibertD

Verbreed je basis! Dat is de opdracht die TestNet me geeft in 2016. Het jaar waarin ik mag vieren dat ik twintig jaar in het testvak zit. En dan zo geconfronteerd worden met het feit dat je niet genoeg weet. Wat doet dat met me?

Een tsunami aan gedachten

Het schudt me opnieuw wakker. Want door de jaren heen heb ik steeds meer geleerd dat ik het eigenlijk allemaal (nog) niet weet. Niet omdat mijn klant of leidinggevende klaagt over mijn kennis, maar meer omdat ik zelf continu wil reflecteren op de effectiviteit en efficiëntie van mijn handelen. Maar net zoals een tester nooit tevreden is over de software, net zomin kan ik tevreden zijn met mijn werk, laat staan mijn kennis en vaardigheden. En daar komt de tsunami aan gedachten al weer aanrollen.

Ik wil meer hands-on ervaring op doen met moderne testtools. Want mijn gedetailleerde kennis van WinRunner 4.02 is inmiddels toch wel gedateerd. Ik moet aan de slag met Open Source tooling zoals Selenium, Fitnesse en Cucumber. Daarnaast wil ik ook wel een upgrade naar HP ALM, gecombineerd met een goede kennis van Performance Center.

Maar het werk gebeurt in DevOps teams met Continuous Integration. Mijn Scrumcursus schiet tekort en ik zet enkele dikke 'agile' boeken op m'n lijst. En Jenkins natuurlijk, waar blijf je anders met je testjes?

Overpeinzingen

Ondertussen knaagt het in me. Testautomatisering is niet de oplossing voor alle problemen. Ik denk terug aan de masterclass Rapid Software Testing en wil in de praktijk meer ervaring opdoen met context-driven testen. Zal ik een mooie Heuristic lijst opstellen en testsessies gaan organiseren met korte en bondige testcharters? Maar hoe gaat dat een eerste keer en willen mijn collega's daar wel in mee? Of zeggen die alleen nog maar: Huh? So? What?

Ah gelukkig, voor het opstellen van de teststrategie van de Mobile devices vind ik op internet een mooie MindMap, alsof ik er heel lang over heb nagedacht.

Na al die jaren automatisering vind ik alle verzamelde informatie terug in de BI-wereld. Een interessant fenomeen waar het gaat over groot, veel en complex. Ben je daar als tester in control? Ja, dat kan en moet maar vergt verdieping. En net als de grip dreigt te komen dan is daar Big Data en alles schudt op z'n grondvesten. Niks meer gestructureerd en alles live-and-streaming.

'Pick your Battles'

Daar zit je dan, terugdenkend aan de eerste testles: een testgeval is een initiële situatie – een actie – een uitkomst, die je vergelijkt met het verwachte resultaat. Maar inmiddels weet ik totaal niet meer wat ik kan verwachten, behalve dat de klant nog steeds kwaliteit verwacht, maar wel sneller en met meer eisen dan voorheen. Joehoe!

Meer vaardigheden? Ja, Time Management! En als dat niet helpt? Cursus Relativeren. En anders?: Pick your Battles. Je kunt nu eenmaal niet alles weten, maar na twintig jaar heeft het testvak nog steeds heel veel moois te bieden. ←

SECOND THOUGHTS ABOUT BLOGGING

Door Han Toan Lim ● info@mindfultester.com @MIndfulTester

2012
'Do you blog?' the job interviewer asked.
'I don't.', I answered.
'That is a shame.'

2015
'Can you tell me, what your strong points are?'
'I have a blog. [...]'

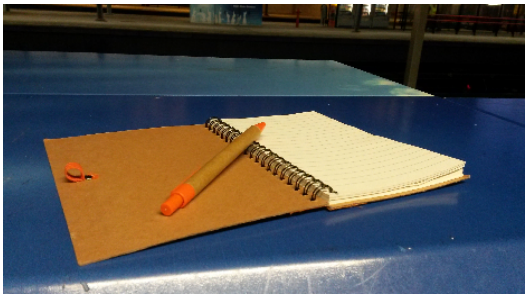
It took me a while to make that step

No tools included

During my career as a tester, I tested several web sites. In some projects I had access to the CMS. I learned to add pages, make links, and add users. The trickiest parts were the connections between the web site and other systems. It could be difficult to set up a good test environment. Especially, if more than one supplier was involved.

However, the work of a web site author is technically not that difficult. Knowledge of a word processor and basic understanding of web sites was enough to make decent web pages. I thought. This was until I had to choose a host and a CMS. It was like someone asked me: 'Do you take this Content Management System as your web blog tool for now and in the future?'

Surprisingly my answer now is 'No.' Every tool has its drawbacks and it is possible to migrate content from one CMS to another.



No certificates needed

During some of my test projects my project members and I had great fun making the most foolish web pages for testing purposes. Pictures of apes and smart comments were a way to enjoy ourselves. The question was: 'Is it possible to add pictures or give comments?'

In the software development industry there are all kinds of certificates. I skipped the whole certification process in order to make web pages for my blog fast. In this case I preferred learning on the job. The preview of a blog post does not ever satisfy me. To me, it still takes some tweaking to make an agreeable web page.

Despite the reasonable maturity of the CMS I did struggle with HTML once. In one particular case I even had to install a plugin to hide my user-id for curious hackers.

Removing Writer's Block

One of the biggest challenges of a blog is to produce posts at a steady pace. At the moment there are more than 200 blogs about software testing. How is it possible, that my blog ended on place 93 of the most popular test blogs in 2015? Writing at least three posts in a blog about testing in 2015.

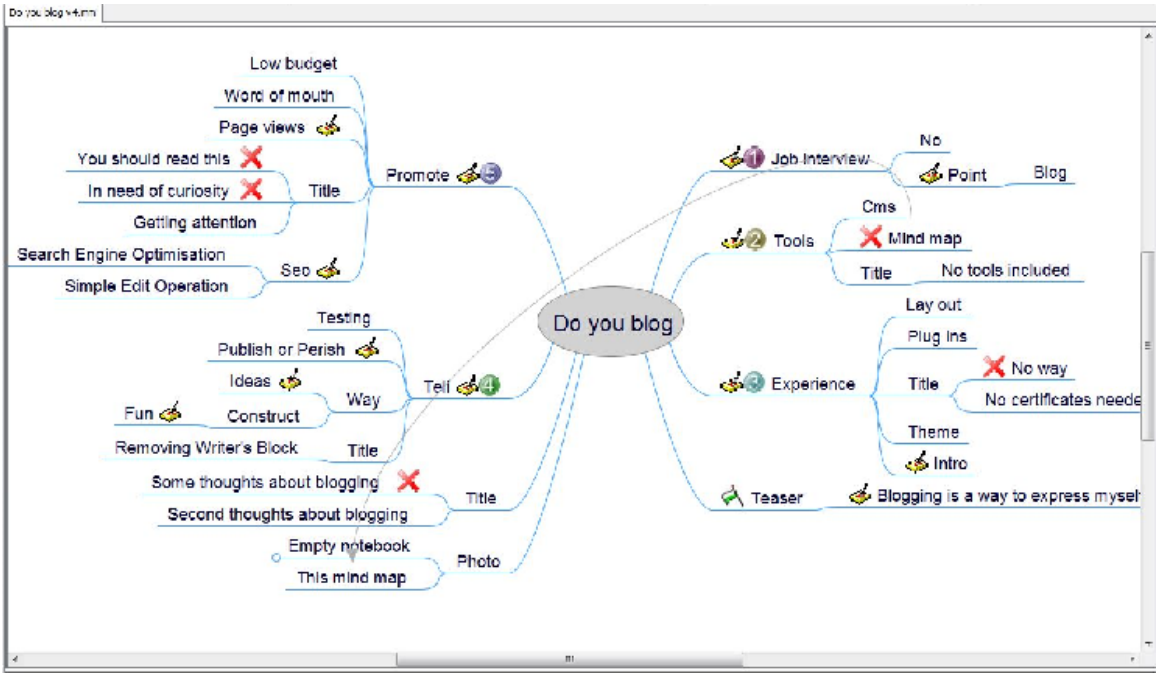
It was sometimes hard to write good blog posts. Especially, if I was preparing for a workshop for a test conference abroad. Almost all my creativity was used for this. Why not blog about the preparations?

My way to create blog posts is to collect blog ideas continuously. My simple question is: Is it bloggable? After blogging about a subject it is possible to do it again using a different point of view. E.g. I blogged several times about testing and juggling. I juggle more than 25 years, so I have enough experience.

Basically a blog idea is an item that you can share. It can be a simple fact or a lengthy story about a challenging phase of a test project. Blog ideas can be combined and enrich each other. Last year I wrote a post containing two juggling stories and one long test story. I added K-cards*) to the writing to give an impression of an actual talk. It was followed by two other blog posts containing Q & A. So I mixed six blog ideas and produced three posts.

A blog idea can also be fun. What about the following headings in one blog post: Dish Cushion, Dish May, Dish Orientation, Dish Count, Dish Way, Dish Missed?

A thought of mine about flow is that I am continuously modifying a post. Even for me it is hard to write down a story in one go, from start to finish, in one hour. It takes me several hours of tweaking to create something for the web. I not only add sentences, but I change and edit at high speed as well, until it is ready to publish. This very story also followed a nonlinear creative process.



Getting attention

The good news was that my new blog was spotted within weeks without any advertising or other money consuming activities. Test blog watchers noticed my blog and used RSS feed for the latest blog posts. The bad thing was that I got commercial mails for hosting and web site redesign.

Of course I had the illusion, that SEO is a Simple Edit Operation in order to get more visitors on my web site. Search Engine Optimisation tools are available, but have some drawbacks I do not like. E.g. the use of cookies leads to the juridical obligation to inform the reader about it.

The trick is to get people, preferably testers or test managers, to go to my test blog. One of the most powerful and free tools within the software testing community on the moment is Twitter. That little Blue Bird provides an easy way to point to my blog posts, which in turn catch the attention of my tweeting peers.

Nog een tip: er zijn weinig Nederlandstalige blogs over softwaretesten.

Happy Blogging!

*) K-cards are cards, used at a conference, to signal the facilitator about joining a discussion ←

ONDERZOEKER VERSUS GEAUTOMATISEERDE CHECK-ROBOT

Door Carlo van Driel ● carlo@deagiletesters.nl

De rol van de tester verandert. Van de veredelde analytische check-robot naar de echte tester: de onderzoeker. De reden is dat, mede door de komst van Agile-Scrum, de handmatige check-robot vervangen kan/moet worden door automatisering. Wat hierbij echter vergeten wordt, is dat daarmee het grootste deel van het werk van een tester dus niet geautomatiseerd kan worden. En hier gaat het op dit moment nog grondig mis in ICT-land...

Vraag naar testautomatiseerders

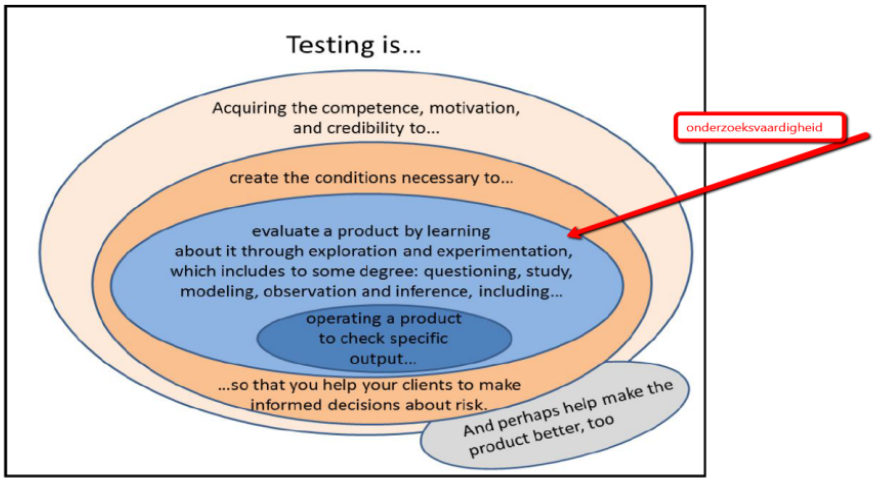
Overal zien we een enorme vraag ontstaan naar testautomatiseerders die de rol van tester moeten overnemen en invullen. Dat is echter de allergrootste fout die je kunt maken. Het automatiseren van checks (en dus niet van testers) is zeker nuttig en zelfs noodzakelijk. Maar het is slechts 25% van het werk dat een tester zou moeten doen. De andere 75% kun je simpelweg niet automatiseren. Deze 75% noemden we tot recent Exploratory (of onderzoekend) testen. Dit omdat zowel het ontstaan van het testgeval alsmede de uitkomst niet vooraf zijn te bepalen. Ik zou er liever voor kiezen om dit gewoon testen te noemen en wat we in het verleden wel testen noemde vanaf nu ´checken´ te noemen. Een onderzoeker volgt geen vaste patronen met vooraf gedefinieerde uitkomsten. Dan is het onderzoek zinloos. Dan bevestig je slechts datgene wat je al weet.

Daarom is de belangrijkste nieuwe vaardigheid voor een tester anno 2016: *onderzoeksvaardigheid*

Vaardigheid

Zoals het woord onderzoeksvaardigheid al aangeeft, betreft het hier een vaardigheid. Geen tooltjes, methoden of trucjes om managers tevreden te stellen met prachtige metrics die de echte werkelijkheid niet weergeven.

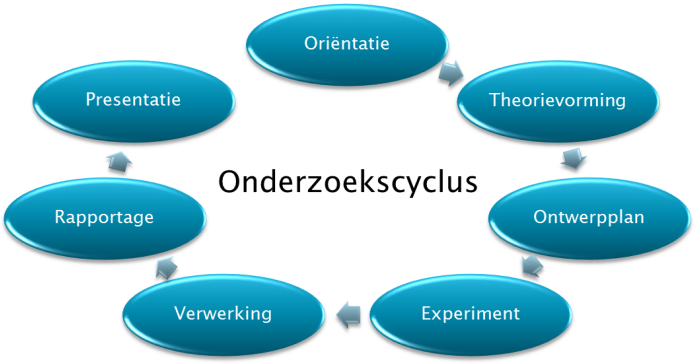
Onderzoeksvaardigheid betekent dat je in staat bent een product te testen door met het daadwerkelijke product te gaan experimenteren, door het te gebruiken en het gedrag van het product te observeren, door modellen te maken van het product en door het product te bevragen. Een nuttig hulpmiddel hierbij zijn heuristics, hierover verderop meer.



Een tester anno 2016 is in staat om het gat te dichten tussen wat we weten en wat we zouden moeten weten om een beslissing te kunnen nemen of het verantwoord is om een product op te leveren naar een klant. Deze beslissing wordt overigens niet genomen door de tester. Deze geeft slechts zo goed mogelijk inzicht in de risico´s. Een tester is ook nooit verantwoordelijk voor de kwaliteit van een product, maar wel voor de kwaliteit van zijn werk.

Onderzoeken

De echte risico´s die de waarde van het product bedreigen, zal een tester moeten onderzoeken. Hij zal moeten experimenteren, vragen stellen, modelleren, observeren en bestuderen. De echte risico´s zitten, zeker in de hedendaagse complexe systemen, vaak daar waar je ze niet ziet en daar waar je ze niet verwacht.



Een product moet gericht onderzocht worden om het onderzoek waarde te geven. Hiervoor zijn ´heuristics´ bijzonder goed bruikbaar. Een heuristic is een feilbare methode om een probleem op te lossen. Feilbaar wil zeggen dat het niet een 100% zekerheid en garantie geeft, maar dat het goed genoeg is. Veelgebruikte heuristics zijn SFDIPOT wat staat voor Structure, Functions, Data, Interfaces, Platform, Operations en Time. Deze heuristics zijn een handvat om je product te testen. De aanpak die ik hanteer, is om een product in een mindmap op basis van deze (en meestal nog vele andere relevante) heuristics te ontrafelen, en van hieruit samen met het team te bepalen waar de risico´s zitten en waar we onze testen op gaan richten. Welke heuristics relevant zijn, is enorm afhankelijk van de aard van het product en zijn omgeving. Tijdens het testen houd je je heuristic en risico´s steeds voor ogen zodat je niet te ver afdwaalt in je testen.

Samenvattend

Beperk je je tot het vervangen van de tester door een set van geautomatiseerde checks, dan zal het gat tussen wat je moet weten over de risico´s van een product en datgene dat je werkelijk weet zeer groot blijven. Daarmee komt direct de klantwaarde van het product in gevaar omdat je maximaal 25% raakt met één heustic (meestal Functions). Over alle andere heuristics kun je geen enkele uitspraak doen en dus is het risico-gat slechts een heel klein beetje kleiner geworden.

Een tester kun je niet vervangen door automatisering, net zo min als dat je testen kunt vervangen door automatisering. Alleen ´checks´ kun je automatiseren. Laten we hopen dat dit besef snel doordringt tot de velen die leven met het beeld dat testen in Agile-Scrum vooral draait om automatisering. ←

TEST DE REST – TESTEN VAN RESTFUL WEBSERVICES MET REST ASSURED

Door Bas Dijkstra ● bas@ontestautomation.com @_basdijkstra

RESTful webservices spelen een grote rol in veel IT-trends, zoals mobile en het Internet of Things. Maar wat is zo'n RESTful webservice nu eigenlijk? En hoe test je zo'n webservice? In dit artikel leg ik dat uit en laat ik zien hoe je met de tool REST Assured eenvoudige en leesbare geautomatiseerde tests voor RESTful webservices kunt opzetten.

IT trends

Mobile, microservices, het Internet of Things, ... Zomaar een paar IT-trends die de afgelopen jaren een vlucht hebben genomen en die waarschijnlijk in de aankomende jaren alleen nog maar een grotere rol gaan spelen. Nog een trend: steeds meer softwareontwikkelaars kiezen ervoor om via API's (delen van) de functionaliteit van hun applicaties te ontsluiten voor de rest van de wereld, zodat in principe iedereen daar gebruik van kan maken bij de ontwikkeling van nieuwe toepassingen. Neem bijvoorbeeld Googles Gmail API of Amazons Simple Storage Service (S3) API. De overeenkomst tussen al deze trends en toepassingen? Overal spelen RESTful webservices een belangrijke rol. Maar wat is een RESTful webservice nu eigenlijk? Hoe kun je als tester tests voor deze webservices opstellen en uitvoeren? En welke tools zou je daarbij kunnen gebruiken?

RESTful webservices uitgelegd

In het kort is een RESTful webservice een webservice op basis van het **RE**presentational **S**tate **T**ransfer-principe. Een dergelijke webservice maakt gebruik van HTTP-requestmethoden en URI's voor het uitvoeren van CRUD-operaties op gegevens. Een eenvoudig voorbeeld ter verduidelijking:

- Een HTTP POST-request naar een URI /users/Tom maakt een nieuwe gebruiker Tom aan
- GET /users/Tom haalt de gegevens van gebruiker Tom op
- PUT /users/Tom doet een update op de gegevens van gebruiker Tom
- DELETE /users/Tom verwijdert Tom uit de lijst van gebruikers

Dit werkt volgens hetzelfde principe als waarmee je browser gegevens van een webserver opvraagt en terugstuurt. Je browser verstuurt bij het laden van een webpagina een GET-request naar de webserver die die pagina host, en hij verstuurt de gegevens die je in een formulier invult via een POST-request naar diezelfde webserver. Een van de belangrijkste verschillen met het ophalen van webpagina's is dat een RESTful webservice gegevens die worden opgehaald meestal in XML- of JSON-formaat teruggeeft, in tegenstelling tot HTML.

Het gaat te ver om hier alle kenmerken en mogelijkheden van REST te beschrijven. Op het internet is veel meer een uitgebreider informatie over het principe achter REST en de verschillende toepassingen te vinden.

Gebruik van RESTful webservices

Er is een aantal goede redenen waarom RESTful webservices worden verkozen boven, bijvoorbeeld, SOAP-gebaseerde webservices:

- REST staat veel verschillende dataformaten toe, waar SOAP bijvoorbeeld alleen maar gebruik kan maken van XML.

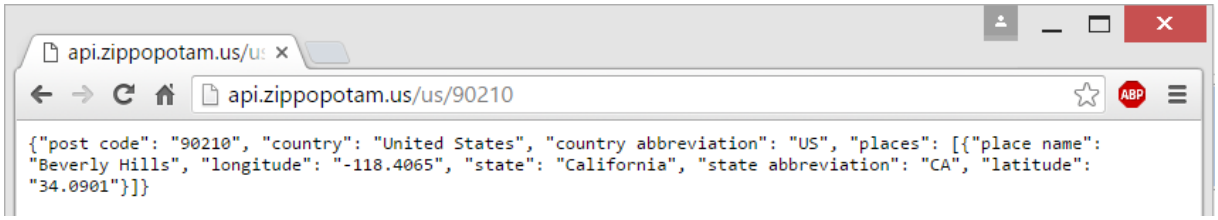
- RESTful webservices hebben een veel kleinere overhead en daarmee een betere performance en schaalbaarheid. Dit is vooral van belang voor mobiele- en IoT-toepassingen.
- REST-calls kunnen worden gecached, waar dat bij SOAP-calls niet mogelijk is. Ook dit heeft een positief effect op de performance.

Aan de andere kant is er ook een aantal zaken die niet door REST worden ondersteund, maar bijvoorbeeld wel door SOAP, zoals WS-Security, WS-ReliableMessaging en WS-AtomicTransaction. Deze mechanismen zijn met name zinvol bij communicatie met en tussen banken, wat dan ook een van de redenen is dat daar nu nog vaak van SOAP gebruik gemaakt wordt. Voor heel veel andere toepassing is REST een eenvoudiger, snellere en steeds meer gekozen optie.

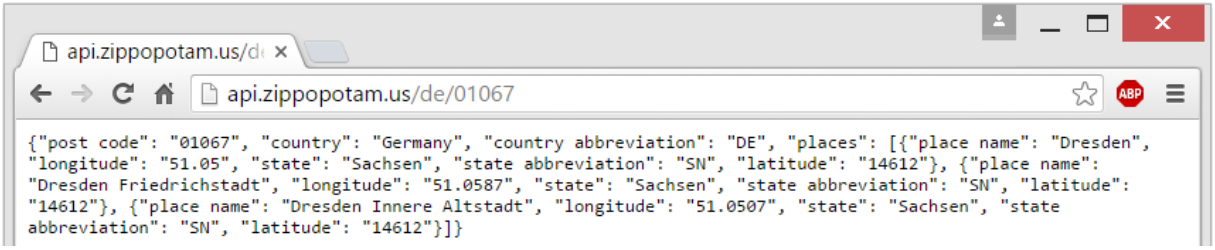
Testen van RESTful webservices

Nu we hebben gezien wat een RESTful webservice is wordt het tijd om eens te gaan kijken hoe je deze services kunt testen.

Voor het uitvoeren van tests op RESTful webservices heb je een testtool nodig die de gewenste call kan uitvoeren. We hebben eerder gezien dat een call naar een RESTful webservice op dezelfde manier wordt uitgevoerd als een request naar een webserver bij het laden van een website. Je kunt een call naar een RESTful webservice dan ook via je browser uitvoeren. Ga bijvoorbeeld maar eens naar http://api.zippopotam.us/us/90210. Dit is gelijk aan het sturen van een GET naar de api.zippopotam.us webservice, waarbij als request parameters de landcode 'us' (voor de Verenigde Staten) en postcode 90210 (voor Beverly Hills) worden meegestuurd. Het resultaat van deze call wordt in JSON-formaat getoond in je browser:



Wanneer je andere parameterwaarden gebruikt, bijvoorbeeld 'de' voor Duitsland en postcode 01067, krijg je uiteraard ook een ander antwoord van de webservice:



Een browser biedt uit zichzelf maar een beperkt aantal mogelijkheden om RESTful webservices te testen. Zo moet je elk request met de hand intypen en is het niet zomaar mogelijk om andere operaties dan GET uit te voeren. Gelukkig bestaan er inmiddels een groot aantal tools om het testen van deze webservices te vergemakkelijken, variërend van:

- Browserplugins zoals Postman voor Chrome, via
- Gratis en open source tools zoals SoapUI en REST Assured (de tool die we hierna in meer detail zullen bekijken), tot
- Enterprise-level tools van leveranciers zoals Parasoft, HP en IBM

In de rest van dit artikel zullen we kijken hoe we REST Assured kunnen gebruiken om REST webservices te testen.

REST Assured

REST Assured is een Java-library die specifiek is ontwikkeld om het testen van RESTful webservices zo eenvoudig mogelijk te maken. De tool is eenvoudig te installeren en configureren: download de laatste versie van de website of gebruik een dependency management-tool zoals Maven om REST Assured als dependency aan een nieuw of een bestaand testproject toe te voegen. Op deze manier kan REST Assured bijvoorbeeld ook heel goed als uitbreiding van bijvoorbeeld een unittestsuite op basis van JUnit of TestNG of juist van een functionele (user interface driven) testsuite op basis van Selenium worden gebruikt.

De kracht van REST Assured is voornamelijk de eenvoud van de syntax. De complexiteit van het opzetten van een connectie, het versturen van een REST-call en het opvangen en valideren van het antwoord wordt grotendeels 'onder water' uitgevoerd. Daarnaast biedt REST Assured een Gherkin-achtige syntax zoals bekend uit Behaviour Driven Development. De onderstaande voorbeeldtest in REST Assured doet hetzelfde als we hierboven handmatig in de browser hebben gedaan: het ophalen van de gegevens die horen bij de postcode 90210 in de Verenigde Staten. Er wordt meteen gecheckt of de variabele country de waarde United States bevat:

```
@Test
public void validateCountry() {

    given().
    when().
        get("http://api.zippopotam.us/us/90210").   // voer een GET-call naar de gewenste URI uit
    then().
        assertThat().
        body("country", equalTo("United States"));  // valideer dat het element 'country' in de body van de response
                                                    // de waarde 'United States' heeft
}
```

Zoals je ziet, is de syntax van REST Assured eenvoudig en goed te lezen, zelfs voor mensen zonder programmeerkennis. De Gherkin-syntax maakt het ook mogelijk om een vorm van BDD te bedrijven, waarbij de uit te voeren testgevallen vooraf worden afgestemd tussen ontwikkelaars, testers en business analisten door het gewenste gedrag van de RESTful webservice te beschrijven op de Given... /When... /Then...-wijze.

Naast één-op-één checks op individuele veldwaarden in het antwoord van de service ondersteunt REST Assured nog een groot aantal andere typen validaties, waaronder:

- Valideren of een responsebericht een bepaald element al dan niet bevat, of een lijst van waarden een bepaalde waarde bevat, en nog veel meer functionele validaties.
- Ondersteunen van het uitvoeren van andere operaties dan de get() die we hiervoor hebben gezien. Zo kun je onder andere post(), put() en delete() gebruiken om andere CRUD-operaties uit te voeren.
- Uitvoeren van controles op de headerinformatie die de webservice terugstuurt. Zo kun je de statuscode die aangeeft of het verzoek goed is verwerkt controleren, en kun je valideren of het antwoord wel zoals afgesproken is in een bepaald MIME-formaat wordt verstuurd.

De onderstaande test laat een voorbeeld van het eerste en het derde type validatie zien:

```
@Test
public void validatePlaceAndStatusCode() {

    given().
    when().
        get("http://api.zippopotam.us/de/01067").                          // voer GET-call uit
    then().
        assertThat().
        statusCode(200).                                                   // valideer HTTP-responsecode = 200 (OK)
        and().
        body("places.\"place name\"", hasItem("Dresden Innere Altstadt"));  // valideer dat 'Dresden Innere Altstadt'
                                                                           // in de lijst van plaatsen met postcode
                                                                           // 01067 zit
}
```

Tot slot

Ik hoop dat dit artikel je iets wijzer heeft gemaakt over wat RESTful webservices zijn en hoe je ze kunt testen. Zoals we hebben kunnen zien, zijn er heel veel mogelijkheden en tools waarmee je je tests kunt uitvoeren, waarvan REST Assured misschien (nog) niet een heel bekende, maar in ieder geval wel een heel krachtige is. ←

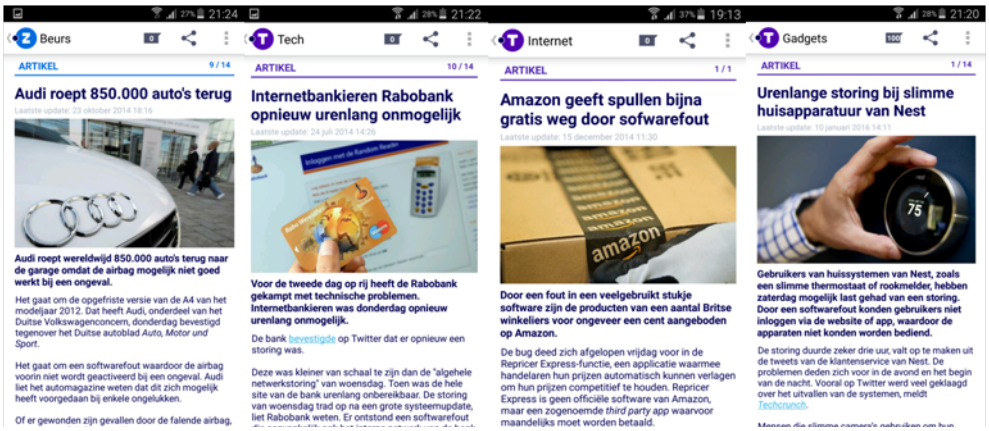
SPECIALISEREN: USABILITY TESTER

Door Erik van Veenendaal ● www.erikvanveenendaal.nl @ErikvVeenendaal

De veranderingen in het IT-landschap hebben ook gevolgen op de rol van tester. De tester heeft feitelijk twee opties: verbreding van kennis en kunde of specialiseren. In dit artikel worden kort beide opties besproken alvorens nader wordt ingegaan op de specialisatie Usability Tester. Het belang van usability zou inmiddels duidelijk moeten zijn in een wereld van mobile apps en websites waarbij veelal de stelling: 'the competion is only click away' van toepassing is. Een aantal technieken wordt aangereikt voor het testen van usability die goed passen in de huidige Agile wereld. Heuristic evaluation, cognitieve walkthrough en vragenlijsten worden behandeld onder de noemer 'Discount Usability Testen'.

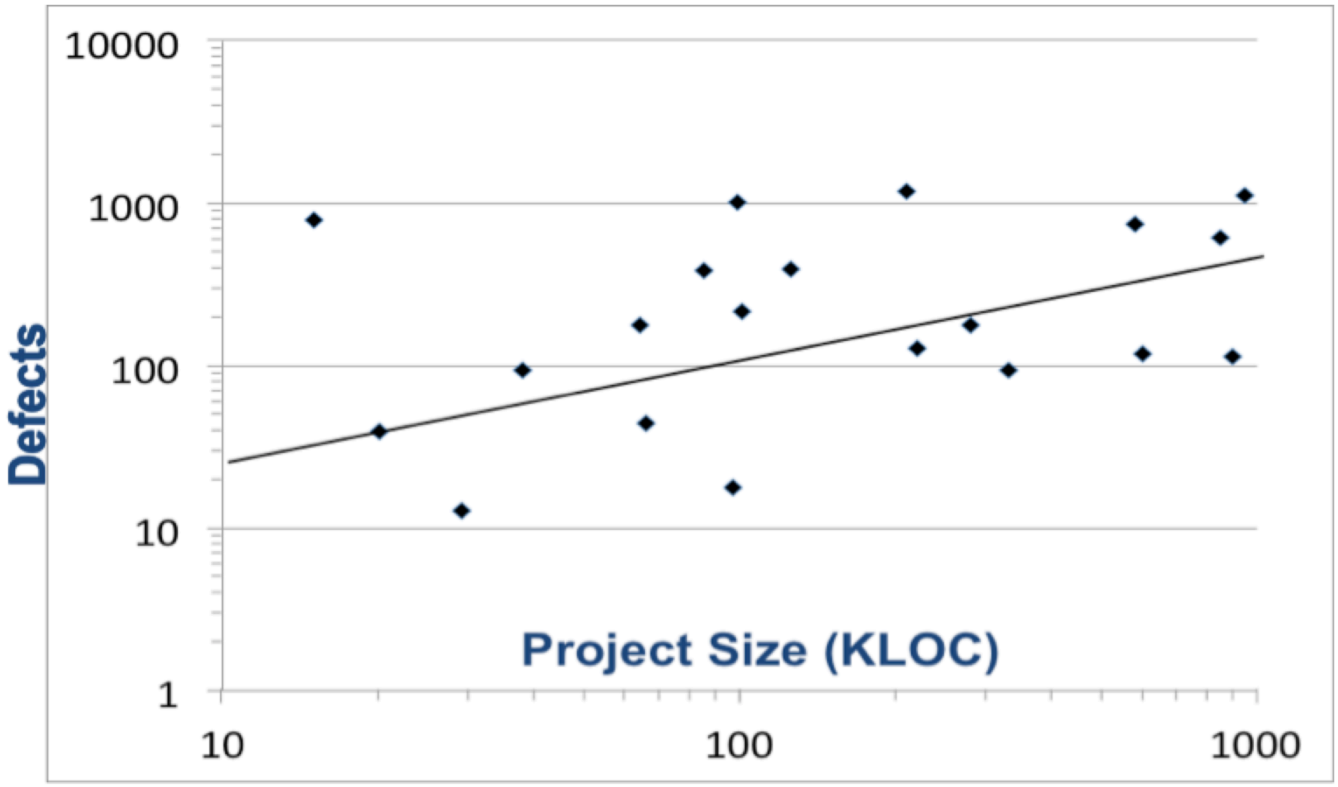
No more Testers!

Met enige regelmaat hoor of lees ik zogenaamde experts verklaren dat binnen een aantal jaren er geen testers meer nodig zullen zijn. Zonder hierover meteen een lange discussie te willen starten, is dit wat mij betreft praktijk versus theorie. In de dagelijkse praktijk zie ik de kwaliteits- en complexiteits-uitdagingen vooralsnog alleen maar groeien. Het aantal fouten is zeker niet aan het afnemen en je hoeft maar een krant open te staan, een nieuws website te bezoeken om weer een verhaal te lezen over de gevolgen van zoveelste software fout (zie figuur 1).



Figuur 1 : Softwarefouten www.nu.nl

Agile heeft allerlei positieve effecten, maar het is een misverstand dat daarmee ineens het kwaliteitsprobleem is opgelost. Een recent Amerikaans onderzoek laat zien dat het aantal fouten per KLOC in een Agile project niet vanzelfsprekend minder zijn, dan bij een traditionele project (zie figuur 2). Ongestructureerd ontwikkelen en testen en dan de transitie maken naar Agile ontwikkelen lost uiteraard niets op in ten aanzien van het kwaliteitsprobleem. Alleen indien tevens stappen worden gemaakt in de volwassenheid van het ontwikkelen en testen, dan pas kunnen verbeteringen worden bereikt. Wat mij betreft nog meer dan voldoende uitdagingen voorlopig voor de tester. Een vergelijkbare conclusie staat ook in het World Quality Report van 2014: 'despite continuous growth in adoption of Agile, many organizations are still facing the the same challenges as in previous years'.



Figuur 2 : Aantal fouten per KLOC Sequentieel vs. Agile
(lijn = sequentiele projecten, punten = Agile projecten)

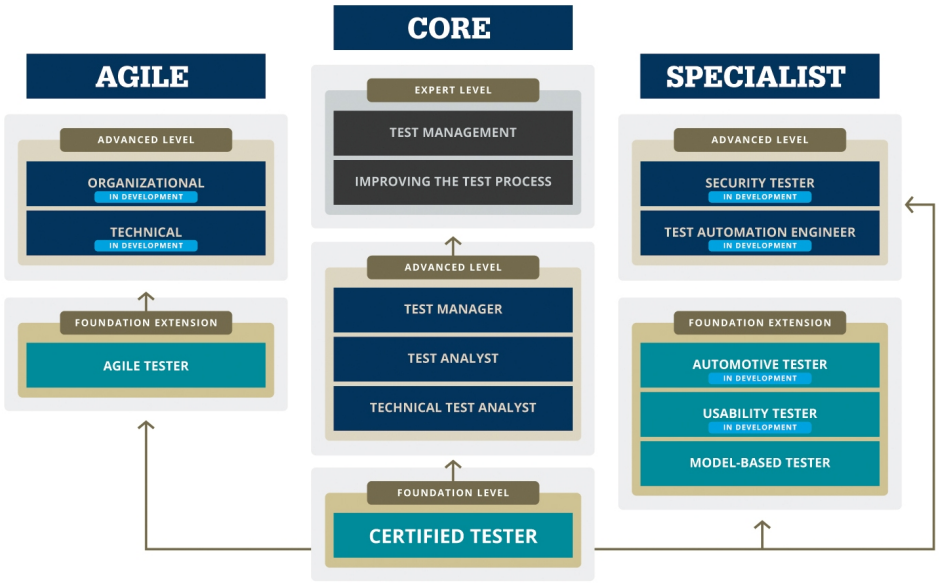
Verbreding van kennis en kunde

Wel is er een echter nadrukkelijk sprake van een verandering van de rol van de tester. De IT-wereld is de laatste jaren sterk veranderd en veel bedrijven zijn, op zijn minst bij een deel van de projecten, overgestapt naar Agile softwareontwikkeling. Een voorbeeld hiervan is dat de tester bij Agile nog meer betrokken bij de requirements dan ooit tevoren. Hij levert een bijdrage aan het specificeren van requirements en de bijbehorende acceptatiecriteria. Zo zijn er diverse trends in het vakgebied testen waarvan de (traditionele) tester op de hoogte dient te zijn en in staat moet zijn hierop te reageren. Kennis en kunde wordt een uitdaging voor veel testers in de nabije toekomst. Het is niet meer voldoende om alleen testkennis en kunde te hebben en alleen te beschikken over een ISTQB of TMap certificaat. Testers zijn veel minder vaak werkzaam in de veilige omgeving van het onafhankelijk testteam. We zullen nauw moeten samenwerken met vertegenwoordigers van de business en ontwikkelaars, elkaar moeten helpen daar waar nodig en als team trachten een product van hoogwaardige kwaliteit op te leveren. Van testers wordt verwacht dat zij onder andere over kennis beschikken betreffende de volgende zaken: domeinkennis, requirements engineering kennis & kunde, kunde ten aanzien van ontwikkelscripts, en sterke sociale vaardigheden, zoals communicatie en onderhandelen. In feite zal de tester zich sterk moeten gaan verbreden ten aanzien van kennis en kunde om ook in de toekomst een goede rol te kunnen vervullen in de veelal Agile projecten.

Specialiseren

Er is echter nog een andere mogelijkheid voor de tester. Aangezien software producten steeds complexer worden en veelal moeten worden geïntegreerd in een open-omgeving, wordt het zogeheten niet-functioneel testen steeds belangrijker maar ook een steeds grotere uitdaging. Tegelijkertijd verwachten de business, de gebruikers en

klanten een hoog kwaliteitsniveau. Hiermee bedoelen ze veelal meer dan alleen de juiste en werkende functionaliteit. Ze willen een systeem dat betrouwbaar is, een goede performance heeft, gebruikt kan worden in diverse omgevingen, uitermate gebruikersvriendelijk is, et cetera. Om in deze uitdagende omgevingen niet-functionele aspecten zoals beveiliging, performance, betrouwbaarheid en usability goed te kunnen testen, zijn specialistische testers nodig. Deze full-time testspecialisten weten 'alles' van het testen van één bepaalde kwaliteitskarakteristiek. Zoals gezegd voorzie ik geen revolutie waarbij de testers zullen verdwijnen. Echter, er zijn trends en we zullen ons (geleidelijk) moeten aanpassen. Als tester heb je de keuze: het verbreden van je kennis en kunde of specialiseren in een bepaald (niet-functioneel) gebied. Interessant in dit opzicht is ook de ontwikkeling binnen het testcertificatieschema van ISTQB. In het zeer onlangs gepubliceerde nieuwe productporfolio (zie figuur 3) zie je een duidelijke stroming gericht op specialisatie.



Figuur 3 : ISTQB Product Portfolio

Eén van de specialistische modules die nu binnen ISTQB volop aandacht krijgt is Usability Tester. Tijdens de general assemblee te Singapore is de bèta versie van de gelijknamige syllabus gereleased. In het vervolg van dit artikel zal ik inzoomen op de usability specialisatie en daarbij een aantal testtechnieken aanreiken die ik zelf als uitermate bruikbaar heb ervaren.

Usability

Het blijft vreemd dat niet minder dan zo'n 40% van de code die wereldwijd wordt ontwikkeld, direct of indirect te relateren is aan de user-interface, dat inmiddels in diverse onderzoeken 'voldoende aandacht voor usability' als een kritieke succesfactor voor IT-projecten is aangeduid, maar we nog steeds slechts een minimaal percentage van de testtijd besteden aan usability testen. Toch zijn er voldoende 'best practices'. In een tijd waarin alles Agile moet worden uitgevoerd met een sterke focus op resultaatgerichtheid; zo min mogelijk documentatie en in een korte tijdsbestek (sprint), denk ik in eerst instantie aan een aantal relatief eenvoudige usability technieken. Dit zijn technieken die snel zijn te leren en vaak niet veel meer kosten dan een één of twee dagen en derhalve een uitstekende kosten-batenverhouding hebben. In dit artikel zal ik nader ingaan op de Heuristic Evaluation, de Cognitieve Walkthrough en de SUMI vragenlijst methode. Deze drie technieken voldoen aan de hierboven gestelde

eisen en passen als zodanige prima in een Agile ontwikkelproject. Een mooie samenvattende term voor deze drie technieken is wat mij betreft 'Discount Usability Testen'. Natuurlijk kan ook gekozen worden voor de traditionele gebruikerstest, maar ook die moet dan anders worden benaderd en meer lean worden uitgevoerd. Vaak blijkt echter een gebruikerstest toch lastiger uit te voeren dan gedacht en ontbreekt de nodig kennis, kunde en ervaring.

Heuristic Evaluation

Heuristic evaluation, ontwikkeld door Jacob Nielsen, is wellicht de meeste bekende en toegepaste techniek om usability te testen. Tijdens een heuristic evaluation wordt de usability van de user-interface systematisch beoordeeld door een aantal usability experts of testers ten opzichte van een aantal usability principes, de zogeheten 'heuristics' (zie tabel 1). Dit kan gebeuren op basis van het user-interface-ontwerp, een prototype of een gerealiseerd systeem. Uiteraard geldt ook hier, hoe eerder hoe beter. De herstelkosten van een fout zullen sterk toenemen indien het usability probleem pas in een laat stadium wordt ontdekt. Het uiteindelijke doel is potentiële usability problemen te vinden (en op te lossen). De heuristics die worden gebruikt, zijn opgesteld op basis van uitgebreid onderzoek naar problemen in de praktijk met usability. De praktijkproblemen zijn in dit onderzoek geanalyseerd en geclusterd. Per cluster is vervolgens een heuristic opgesteld waarmee de problemen (deels) hadden kunnen worden voorkomen. Veruit de meeste problemen in de praktijk blijken terug te herleiden zijn tot het niet voldoen aan deze elementaire usability principes.

1	Zichtbaarheid van de systeemstatus
2	Overeenkomst tussen systeem en de echte wereld
3	Controle en vrijheid voor de gebruiker
4	Consistentie en voldoen standaards
5	Foutpreventie
6	Herkenning in plaats van herinneren
7	Flexibiliteit en efficiëntie van gebruik
8	Esthetisch en minimalistisch ontwerp
9	Hulp aan gebruiker om fouten te herkennen, oorzaak te achterhalen en te herstellen
10	Help en documentatie

Tabel 1 : Tien heurtistic van Jacob Nielsen

De heuristic evaluation is gebaseerd op software-inspectietechniek. Nielsen heeft de inspectietechniek aangepast en geschikt gemaakt om te kunnen gebruiken bij usabilitytesten. In het algemeen kun je stellen dat een heuristic evaluation lastig effectief is uit te voeren door slechts één persoon, aangezien die persoon nooit alle usabilityproblemen van de user-interface zal vinden. Praktijkervaringen in een groot aantal projecten hebben aangetoond dat verschillende testers veelal verschillende usabilityproblemen vinden. Het is daarom mogelijk de effectiviteit van de techniek aanmerkelijk te verbeteren door er meerdere testers (en wellicht ook ontwikkelaars, usability-experts en gebruikers) bij te betrekken. De effectiviteit neemt nog verder toe indien aan de verschillende testers verschillende rollen (op basis van de heuristics) worden toegewezen. Ervaren teams zijn zelfs in staat zo'n 100 usability problemen per uur te vinden.

Tijdens de heuristic evaluation inspecteert iedere tester de interface zelfstandig. Slechts nadat alle individuele evaluaties zijn afgerond is het toegestaan aan de testers om onderling te communiceren en de bevindingen samen

te voegen of te clusteren. Dit aspect is belangrijk en dient ervoor te zorgen dat de evaluaties onafhankelijk zijn en er geen onderlinge beïnvloeding plaatsvindt. Een heuristic evaluation sessie duurt normaal 1 à 2 uur. Langere sessies zijn soms nodig indien de user-interface erg complex of omvangrijk is met een groot aantal schermen en opties. Het is echter aan te raden om dit soort sessies dan op te splitsen in meerdere kortere sessies en eventueel te verdelen over meerdere personen. Het blijkt namelijk dat na zo'n twee uur de concentratie afneemt en er minder problemen worden gevonden. Tijdens een sessie zal de tester normaal gesproken meerdere keren door de interface heen gaan en het beoordelen ten opzichte van de heuristics. Hierbij wordt soms gebruikgemaakt van een checklist die veel voorkomende fouten aangeeft per heurstic. De heuristics worden gebruik om fouten te vinden, maar veelal ook om de gevonden problemen te categoriseren.

In principe bepaalt de tester zelf hoe hij de evaluatie uitvoert. Een algemene richtlijn is wel dat de tester tenminste tweemaal door de user-interface heen gaat. De eerste keer is het doel inzicht te krijgen van de procesgang binnen de interface en kennis te maken met het systeem. Tijdens de tweede doorgang kan de tester dan de verschillende interface elementen beoordelen daarbij wetende hoe zij passen in het groter geheel. Zoals gezegd kan de heuristic evaluation uitstekend worden toegepast in een vroegtijdig stadium van ontwikkeling. De praktijk heeft aangetoond dat deze techniek, die vrij eenvoudig te leren is, een uitermate efficiënte usability testtechniek is.

Cognitieve Walkthrough

De cognitive walkthrough is een usability reviewtechniek die begint met het opstellen van een aantal gebruikers-scenario's op basis van de requirements (user-stories) en/of een prototype. In tegenstelling tot de heuristic evaluation staan bij de cognitieve walkthrough de taken van de gebruiker centraal en wordt vanuit de taken getest. Je zult dus ook veelal andere fouten vinden dan bij een heuristic evaluation. Met de taakgerichte aanpak wordt getracht zoveel mogelijk fouten te vinden die ook in de praktijk kunnen optreden. De cognitieve walkthrough is derhalve een taakgerichte testtechniek die wordt uitgevoerd door een aantal usability experts / testers om op die manier fouten in het ontwerp van de user-interface te ontdekken.

De usability tester speelt feitelijk de rol van de gebruiker waarbij hij door de user-interface heen gaat en probeert de gebruikerstaak uit te voeren. Elke stap binnen de taak die de gebruiker zou moeten nemen wordt uitgebreid geëvalueerd. Als de usability tester vastloopt in de user-interface of de taak niet (volledig) kan uitvoeren duidt dit op een probleem. Wellicht dient de interface eenduidiger en eenvoudiger worden gemaakt. De analysefase bestaat uit het beoordelen van elke stap binnen de taak en telkens trachten te bedenken waarom de gebruiker de juiste actie zal kiezen (of juist niet). Deze analyse zijn mede gebaseerd op kennis en aannames over de achtergrond en doelstellingen van de gebruiker, maar ook het begrijpen wat het mentale proces is wat de gebruiker doormaakt bij het kiezen van de juiste actie.

Tijdens de cognitieve walkthrough stelt de tester zich telkens de volgende vier vragen:

1. Is het voor de gebruikers duidelijk welke actie hij moet uitvoeren?
   Bijvoorbeeld is de taak het printen van een document, waarbij eerst een printer moet worden geselecteerd. Weet de gebruiker dat hij dat eerst een printer moet selecteren?
2. Wordt de beschikbaarheid van betreffende actie door de gebruiker opgemerkt?
   Dit heeft te maken met de zichtbaarheid en begrijpbaarheid van de mogelijkheden in de user-interface.
3. Zal de gebruiker de juiste actie associëren met het resultaat dat hij wil bereiken?

4. Als de juiste actie is uitgevoerd, is het voor de gebruiker duidelijk dat er voortgang is in het afronden van de actie?
   Dit heeft te maken met het beoordelen van de feedback van het systeem nadat de gebruiker de actie heeft uitgevoerd.

In de praktijk blijkt dat de walkthrough techniek ietwat lastiger te leren is dan bijvoorbeeld de heuristic evaluation en men ook meer ervaring moet hebben om deze goed te kunnen uitvoeren. Kennis van zowel de gebruikers van het systeem, maar ook algemene kennis en ervaring ten aanzien van het gedrag van gebruikers is noodzakelijk. Ook een cognitive walkthrough kan in elk stadium van de ontwikkeling van het ontwerp worden uitgevoerd, waarbij er gebruik kan worden gemaakt van een prototype of het reeds gerealiseerde systeem.

SUMI vragenlijst

Een hele andere wijze van beoordelen is gebruikmakend van vragenlijsten. Met behulp van vragenlijsten wordt de mening van gebruikers gevraagd over de usability van het systeem. Hoewel in principe ook toepasbaar op prototypes of zelfs een user-interface-ontwerp document, worden vragenlijsten met name toegepast wanneer het systeem in (acceptatie)test is of als het in productie is. Dit is meteen een groot verschil (en nadeel) met de twee voorgaande testtechnieken. Daar waar de heuristic evaluation en cognitieve walkthrough al in een vroegtijdig stadium van de ontwikkeling kunnen worden toegepast, zal een vragenlijst in het algemeen pas in een relatief late fase van de ontwikkeling kunnen worden uitgevoerd. Soms wordt een vragenlijst gebruikt om de mening van acceptatietesters en gebruikers te krijgen over usability, tijdens of na afloop van een testfase.

Zodra er voldoende gebruikers de vragenlijst hebben ingevuld om er statistisch verantwoorde uitspraken mee te kunnen doen, volgt de verwerking en vervolgens de analyse van de resultaten. Hoewel de vragenlijst techniek een relatief goedkope manier is om usability te testen dan wel inzicht te krijgen in de usability van een user-interface, is een belangrijk ander nadeel dat de vragenlijsttechniek normaal gesproken geen gedetailleerde bevindingen oplevert. Als bijvoorbeeld de usability als onvoldoende wordt beoordeeld, zullen aanvullende testen moeten worden uitgevoerd om de concrete problemen op te sporen.

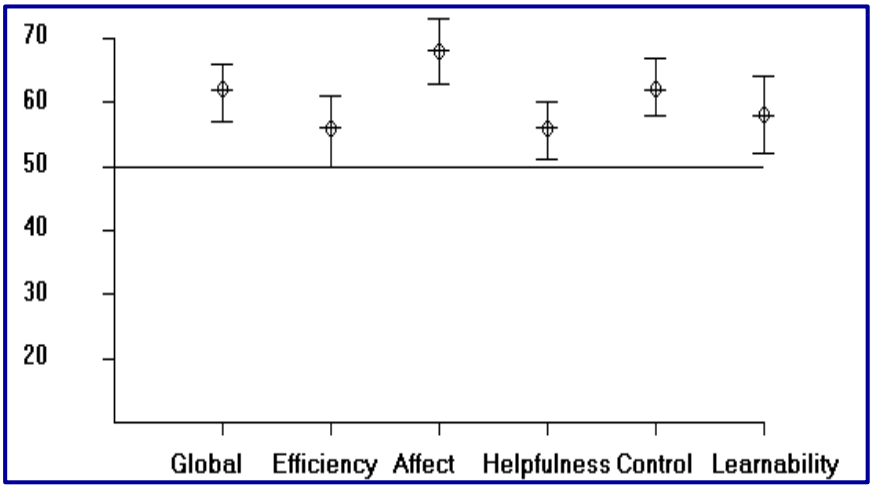
Natuurlijk kan men zelf een vragenlijst ontwikkelen, maar het is aan te raden gebruik te maken van een standaard vragenlijst. Het ontwikkelen van een onderbouwde en uitgebalanceerde vragenlijst met goede vragen is niet zo makkelijk als het wellicht lijkt. Bekende voorbeelden van veelgebruikte standaard vragenlijsten zijn onder andere SUMI (Software Usability Measurment Inventory) en WAMMI (Website Analysis en MeasureMent Inventory). In het vervolg van het artikel zal ik met name ingaan op de SUMI vragenlijst, WAMMI volgt feitelijk dezelfde structuur en opzet als SUMI, maar is specifiek ontwikkeld voor websites.

In principe is SUMI bruikbaar voor elk software product dat een user-interface heeft. Bij de SUMI vragenlijst wordt usability gemeten ten opzichte van een referentie database. Deze vertegenwoordigt als-het-ware de usability zoals gebruikers deze ervaren in hun dagelijkse praktijk. SUMI bestaat uit vijftig vragen waarbij elke stelling moet worden beantwoord met ‘eens’, ‘geen oordeel’ of ‘oneens’. Hieronder een aantal voorbeelden van de stellingen waarop de gebruiker moet reageren:

- Dit product reageert te traag op de invoer van de gebruiker;
- Ik zou mijn collega's aanraden dit product te gebruiken;
- Het product is er op een bepaald tijdstip onverwachts mee gestopt;
- Het leren omgaan met het product levert in het begin veel problemen op;

- Met dit product weet ik vaak helemaal niet wat het volgende is dat ik moet doen;
- Ik vind het werken met dit product leuk;
- De berichten van het product zijn niet erg behulpzaam.

Zowel de SUMI als WAMMi vragenlijst zijn overigens beschikbaar in een groot aantal talen waaronder Nederlands. De verwerking van de gegevens levert bij SUMI een algemene score op ten aanzien van usabilty, maar ook een vijftal subscores ten aanzien van een aantal usability aspecten (zie figuur 4). Dit is uiteraard erg behulpzaam bij het analyseren van het resultaat. De scores worden bij SUMI berekend door een vergelijking te maken met de reeds eerdergenoemde referentiedatabase van eerder uitgevoerde SUMI-tests op allerlei soorten software producten.



Figuur 4 : Voorbeeld van de SUMI score grafiek

Figuur 4 geeft een voorbeeld van de output die wordt verkregen na een SUMI-test (waarbij gebruik is gemaakt van het ondersteunde tool). De scores van de test en bandbreedte van de scores (bepaald op basis van de standaard deviatie) worden afgezet tegen de gemiddelde score in de referentie database (weergegeven door de waarde 50). De usability scores zoals deze worden weergegeven in dit voorbeeld zijn derhalve positief; dit wil zeggen hoger dan de gemiddelde usability score en met een relatief kleine bandbreedte (weergegeven door de verticale lijn met een boven- en ondergrens).

Een productevaluatie met de SUMI vragenlijst geeft een duidelijke en objectieve meting van de mening van de gebruiker over de usability van het product voor zijn of haar taken. Het kwantitatieve karakter van SUMI is uiteraard een erg sterke eigenschap; usability wordt hiermee meetbaar gemaakt. Indien de score niet conform verwachting is, zullen echter aanvullende tests moeten worden gedaan om de details van de problemen te achterhalen. Soms wordt, om dit te ondervangen, een SUMI-test gecombineerd met een heuristic evaluation of cognitieve walkthrough waardoor men zowel een kwantitatieve usability meting heeft alsmede een gedetailleerde lijst met usability bevindingen.

Ten slotte

Reeds een groot aantal jaren verzorg ik cursussen en spreek ik op (test)conferenties over het onderwerp Usability. Tijdens EuroSTAR heb ik al eens een ‘award-winning’ tutorial verzorgd met als onderwerp ‘Discount Usability Testing’. Toch heb ik niet het idee dat het baanbrekend werk al heel veel heeft opgeleverd. Misschien waren we er

nog niet klaar voor, of was gewoon de behoefte er nog niet (in voldoende mate). Met de beweging naar meer specialisaties binnen het vakgebied testen, komen er wellicht usabilitytesters die dit verder gaan uitdragen en oppakken. In deze tijd van mobile, internet, et cetera zou je toch haast denken dat het niet anders kan. Tijdens een recente conferentie werd ik een aantal keren aangesproken door enthousiaste ex-cursisten die vertelden dat ze de bovengenoemde technieken met uitstekende resultaten nu echt gebruiken in hun projecten. Gaat het dan toch nog gebeuren?! ←

DE TESTER, EEN VRIENDELIJKE ONDERZOEKDE ACTIVIST

Door Rik Teuben ● rik.teuben@bjtesting.nl

Als tester doe je onderzoek naar systeemkwaliteit. Vraag je jezelf ook wel eens af waarom je dat onderzoek eigenlijk allemaal doet? Doet het ertoe dat we ondanks de dagelijkse standup, de productdemo's, geverifieerde specs en acceptatiecriteria uiteindelijk moeten vaststellen dat één of meerdere stakeholders het nieuwe systeem of de systeemaanpassingen met skepsis en teleurstelling ontvangt. Inzicht geven in de systeemkwaliteit is natuurlijk van zichzelf al een leuke dagbesteding, maar écht leuk wordt het pas als je als tester niet alleen de systeemkwaliteit maar ook de acceptatie door de belanghebbenden actief begeleidt en gezamenlijk met hen, de skepsis vertaalt in onderzoeksdoelen?

Actie-onderzoek

Gewoonlijk stel je jezelf als tester terughoudend op als het aankomt op de vraag wat er moet gebeuren met de informatie die je aanlevert. Je signaleert en rapporteert, maar wat er met de informatie gebeurt en wat er gedaan moet worden is aan de anderen. In de gammawetenschap[1] (ons volwassen onderzoeksbroertje waar we als testers minstens zoveel overeenkomsten mee hebben als met de bèta-wetenschappen) heeft deze afstandelijke onderzoekshouding geleid tot een ge-ëngageerde tegenbeweging. Deze staat bekend onder de naam Participatief Actie Onderzoek (PAO). Sinds het laatste deel van de vorige eeuw is het een belangrijke, niet meer weg te denken stroming geworden. PAO vult de lacune in die menig gamma-onderzoeker ervaart als hij zich moet redden met de onderzoeksmethoden die uit de bèta-faculteiten worden aangeboden. Of om het concreet naar onze eigen testpraktijk te vertalen; de rationeel analytische testaanpak is prachtig, maar schiet tekort zodra we de verificatietesten ('technology faced') achter ons hebben gelaten en ons bezig gaan houden met waar het echt om draait: de tevredenheid van de belanghebbenden.

'Testers zouden vaker "geitenwollen sokken" moeten dragen onder hun "witte jas".'

Kenmerkend voor Participatief Actie Onderzoek is:

- Cyclisch – soortgelijke stappen herhalen zich in gelijke volgorde;
- Participatief – Onderzoekers en belanghebbenden werken samen als onderzoekspartners en participeren gelijkwaardig in het onderzoek;
- Kwalitatief – Resultaten worden vaker gerapporteerd in tekst dan in grafieken en getallenoverzichten;
- Reflectief – Kritische reflectie op het onderzoeksproces en op de uitkomsten ervan is een belangrijk onderdeel van de cyclus.

Soortgelijke taakopvattingen zie je steeds vaker terug bij 'opinion leaders' binnen ons vak. Intelligente, 'niet op hun mond gevallen' actieve heren als Kaner, Bach en Bolton zetten zich af tegen het establishement en roeren thema's aan die in de jaren zeventig ook op de agenda stonden van de groep onderzoekers die zich zich als Action

[1] *Met **gammawetenschappen** bedoelt men in Nederland en Vlaanderen de wetenschappen die zich met maatschappij en gedrag bezighouden: onder andere sociologie, antropologie, economie, rechten, politicologie, psychologie en communicatiewetenschap.*

Researchers profileerde. Het belang van exploratie, het gevaar van onderzoeksbias (je ziet wat je wil zien), en de beperkingen van 'denken in modellen' zijn maar enkele illustraties van de overeenkomsten tussen de denkbeelden van de bevlogen groep Action Reserachers en de groep testers die zich als tegenbeweging profileert onder de naam Context- Driven Testers. Tijdens mijn 'spreekbeurt' op het voorjaarsevenement zullen we, nadat we wat meer bekend zijn geworden met PAO, gezamenlijk inventariseren of er nog andere thema's te adresseren zijn die eerder door de groep van actie-onderzoekers is gesignaleerd. Misschien morrelt dan de onderzoeksopvattingen van de onderzoeksactivisten uit de wetenschap ook wel aan jouw gestolde testopvattingen.

Doen jouw bevindingen er toe?

Als je naar het testvak kijkt als Participatief Actie-onderzoeker, dan zul je vaststellen dat net als in het traditioneel wetenschappelijk onderzoek de impact van testen vaak erg beperkt is. Een te rationele benadering van het testvak, waarin het genereren van bevindingen of het blootleggen risico's ons hoogste doel is, ligt aan de basis van deze beperkingen. Het idee dat veranderingen het meest effectief worden ondersteund met een lijst van bevindingen en openstaande risico's is een tamelijk naïeve opvatting.

Als je met beide benen op de grond staat, dan weet je natuurlijk dat die bevindingen slechts één van de vele inputs vormen in het nemen van projectbeslissingen. Politieke- en financiële haalbaarheid zijn naast tijdbeperkingen vaak veel meer doorslaggevend bij het nemen van besluiten over het wel of niet honoreren van aangescherpte systeemeisen en herstel van fouten. De interventiekracht van een bevindingenlijst en risicolijst wordt doorgaans overschat of kunstmatig hoog gehouden.

> 'Een testproces dat de kenmerken heeft van Participatief Actie Onderzoek levert meer Impact en Value dan een zuiver analytisch testproces.'
>
> *Uit: Opleidingsmateriaal P/Pet*

Met de middelen die ons in projectverband worden aangereikt, ongeacht of het nu een Agile of waterval project is, mag het een klein wondertje heten dat we soms toch de vinger op de werkelijk zere plekken leggen. Door te kiezen voor een testaanpak die de kenmerken draagt van Participatief Actie Onderzoek, vergroot je niet alleen de kans op relevante bevindingen, maar ook de impact van je werk en de waarde van jouw testactiviteiten. En dat is goed voor alle betrokkenen.

Samenvattend

Ons vakgebied wordt vaak vergeleken met het doen van onderzoek. Daarbij is de gangbare opvatting dat goed onderzoek in isolement en door een expert wordt uitgevoerd. Resultaten worden in kengetallen en grafieken aan het publiek gepresenteerd, en de onderzoeker is 'in the lead'. Nog steeds denken ook sommige testprofessionals dat de tester 'in the lead' is bij de uitvoering van testprocessen. Forget it! Die opvatting is in rap tempo op zijn retour bij projectmanagers en ICT opinion leaders. Dat hoeft echter niet persé slecht voor ons als testprofessionals uit te pakken. Integendeel! In dat verband kunnen we nog flink wat leren van de groep softe sociaalwetenschappers die zich profileren als Actie Onderzoekers. Durf jij ook geitenwollen sokken onder je witte onderzoekersjas te dragen? … Jawel toch…? Nou hup, in actie dan! ←

JAPANSE WIJSHEDEN VOOR DE TESTPROFESSIONAL

Door Berry Kersten ● berry.kersten@improveqs.nl @berrykersten

In de loop van de tijd doe je veel nieuwe kennis en vaardigheden op. Maar op een gegeven moment kun je het gevoel hebben dat je leercurve niet meer zo snel stijgt als je zou willen. Dat is jammer, want blijven leren is een belangrijke katalysator voor jezelf, zeker op een Agile project.

Maar om in jezelf te blijven investeren, moet je wel weten hoe! Een Japanse visie kan hierbij nieuwe inzichten geven.

Vanuit mijn interesse voor het Japanse gedachtegoed over kennismanagement geef ik in dit artikel op een niet-alledaagse manier aandacht aan alledaagse zaken. Door middel van een aantal Japanse termen wordt inzicht gegeven in het belang van kennis, leren, verbeteren en herkennen van je kwaliteiten.

Agile Manifesto

Agile kan worden beschouwd als een interactiemodel om wendbaar te worden. De waarden en principes hieromtrent zijn verwoord in het Agile Manifesto (ref 1). De eerste zin van het manifest is: '**we are uncovering better ways of developing software by doing it and helping others do it**'.

Zelf vind ik 'we are uncovering better ways' treffend. Want dit impliceert een proces van leren en verbeteren. Alvorens dit proces te doorlopen is het goed eerst eens een stap terug te doen en jezelf af te vragen waarom je dit allemaal zou moeten doen. Dit kan onder andere door te kijken naar het Japanse begrip Ikigai.

Ikigai

Voor de Japanners is Ikigai een onderdeel van het dagelijks leven. Het betekent: 'de reden waarom je elke dag opstaat'. Ikigai is een combinatie van spirituele elementen, zoals missie en roeping, en praktische en sociale elementen als wat de wereld nodig heeft en waar je voor betaald wordt. Dit wordt weergegeven in de afbeelding.

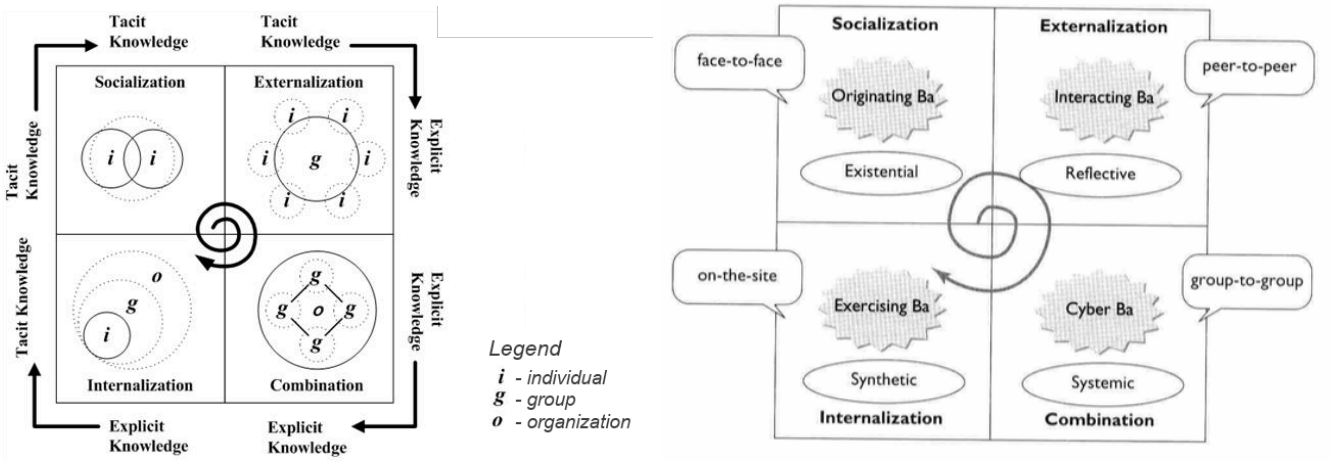
Iedereen heeft Ikigai, maar het is een zaak van bewustwording en keuzes maken. Sommigen kunnen dit ervaren als zweverig, maar je kunt het ook beschouwen als een alternatief voor bestaande theorieën over behoeften en motivatie. Denk aan Daniel Pink's Drive (ref 2) of de behoeftepiramide van Maslow (ref 3).

Lijkt het je de moeite om te onderzoeken hoe Ikigai bij jou in elkaar steekt? Vul je eigen Ikigai-schema in en kijk hoe je er nu voor staat en vul het nog eens in voor hoe je wilt dat het er over een jaar voor staat.

Uiteindelijk kun je als kenniswerker met Ikigai doelgerichter met je carrièrepad omgaan, want kennis is van belang om jezelf wendbaar te maken in een snel veranderende wereld. Volgens de Japanse professoren Ikujiro Nonaka en Hirotaka Takeuchi is menselijke interactie een randvoorwaarde om nieuwe kennis te creëren (ref 4).

Ba

Op basis van onderzoek in succesvolle Japanse bedrijven, realiseerden Nonaka en Takeuchi een theorie voor het ontwikkelen van kennis in organisaties, te weten het SECI-model: Socialisatie, Externalisatie, Combinatie en Internalisatie. Of, de stroom van kenniscreatie waarin spiraalsgewijs impliciete kennis in expliciete kennis wordt omgezet en vice versa. De ruimte en de context waarin deze kruisbestuiving van kennis plaatsvindt wordt Ba genoemd. Deze ruimte is niet alleen fysiek, maar ook mentaal en virtueel. Ba kan dan een platform zijn voor het bevorderen van individuele en/of collectieve kennis op de werkvloer. Nonaka en Takeuchi visualiseren SECI en Ba als volgt.



Een mond vol, maar wat betekent dit in de praktijk? In termen van Scrum reflecteert Ba de processen om kennis te delen, te ontwikkelen en om te leren. Volgens Fægri (ref 5) zijn deze processen zeer voornaam bij de hele cyclus van softwareontwikkeling. De concepten van Ba en SECI zijn complex en het gaat in dit artikel te ver om deze theorie diep uit te werken, maar mijn stelling is dat het nuttig is om je bewust te zijn van deze concepten. Verandering van kennis en vaardigheden begint tenslotte bij bewustwording.

Om verder in te zoomen naar de praktijk geef ik wat voorbeelden (ref 7 & 8) vanuit Scrum. Want het huidige Scrum raamwerk is gebaseerd op de theorie van het werk van Nonaka en Takeuchi (ref 9).

Socialisatie

Het delen van impliciete kennis tussen teamleden tijdens de stand-up. In deze Originating Ba worden ervaringen, gevoelens en emoties uitgewisseld.

Externalisatie
Het expliciet maken van impliciete kennis via dialoog. Het team bepaalt wat en in hoeverre iets gedocumenteerd wordt in user stories via de Interacting Ba.

Combinatie
Expliciete kennis uit verschillende bronnen worden samengevoegd en geordend. Dit leidt tot nieuwe expliciete kennis. Denk aan de BDD-aanpak 'Specification by Example' (ref 10) in combinatie met Cucumber (ref 11) of SpecFlow (ref 12). Dit is een virtuele (Cyber) Ba waarin gebruik wordt gemaakt van tooling.

Internalisatie
Expliciete kennis wordt impliciet gemaakt. In feite leren door te doen waar de opgedane kennis eigen wordt gemaakt en men bekwamer wordt. Ik zie dit als een 'kennis increment' nadat een sprint afgelopen is in de Excercising Ba. Tijdens mijn opgedane Agile ervaring de afgelopen jaren heb ik mijn eigen kennis increment opgebouwd en heb dit vertaald naar nieuwe vaardigheden op bijvoorbeeld het vlak van testautomatisering en requirements engineering.

SECI en Ba hebben mij geholpen om een betere T-shaped testprofessional te worden. De context (Scrum) heeft dit gefaciliteerd. Andersom heeft een Scrum team professionals nodig die daarin weten te acteren. Om deze interactie succesvol te laten zijn, zijn gecommitteerde mensen nodig die willen leren en verbeteren. Dit laatste wordt ook wel Kaizen genoemd.

Kaizen

Beter worden door veranderen via kleine stappen; dat is in een notendop wat de Japanse filosofie Kaizen inhoudt. Het Japanse woord 'kai' betekent veranderen en 'zen' betekent 'goed'. Kaizen is oorspronkelijk gericht op het continu verbeteren van organisatieprocessen, maar de filosofie is prima toepasbaar op het individu. Veelal komen deze inzichten vanuit de mindfulness hoek. Persoonlijk interpreteer ik Kaizen als meer rendement uit jezelf halen door ononderbroken verandering en verbetering. Dit klinkt als een utopie, maar vaak is het een kwestie van gewoon doen en focus blijven houden.

In een blog van Blaz Koz (ref 13) staan enkele tips die kunnen helpen:

Onderzoek je talenten. Je hebt vaardigheden die je kunt oppoetsen, maar het is zaak je ware talent(en) en passie(s) te vinden waarin je kunt excelleren. Zelf heb ik via een cursus 'Personal Branding' ontdekt waar mijn eigen talenten liggen.

De omgeving is belangrijk. We zijn deels een 'product' van de omgeving. Zoek de omgeving (werkgever, opdracht, netwerk) waarin je je kunt profileren.

Er zijn altijd kansen. Je hebt altijd een keuze om jezelf beter te maken. Neem een proactieve houding aan en ga uitdagingen aan.

Maak kleine stappen. Soms lijkt een verbeterdoel groot, maar je kunt (net als op de werkvloer) uitdagingen opbreken in kleinere brokken.

Let op je valkuilen. Tijdens het proces van verbeteren kun je problemen tegenkomen, teleurgesteld worden of fouten maken. Leer hiervan en bewandel een ander pad zodat je niet in dezelfde valkuilen stapt als voorheen.

Een andere manier om jezelf te verbeteren is vaak oefenen, want oefening baart kunst. In Japan noemen ze dit Kata.

Kata

In de Japanse industrie is Kata 'de dingen op de juiste manier blijven doen'. Kata is tevens een term bij karate: een patroon van bewegingen waarin series van technieken zitten. Deze bewegingen worden heel vaak herhaald om stap-voor-stap te verbeteren. Bij eXtreme Programming is het begrip Code Kata (ref 14) niet onbekend. Het doel hiervan is simpelweg een betere softwareontwikkelaar te worden door vaak te oefenen. Wil je als tester soortgelijke kata's uitvoeren? Neem dan eens een kijkje in de Testing Dojo (ref 15).

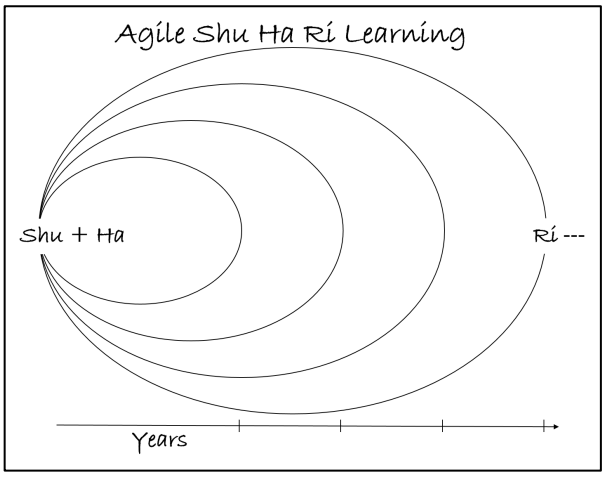
Vanuit de Japanse vechtsport Aikido is er nog een soortgelijke term die ik interessant vind, namelijk Shuhari.

ShuHaRi

Shuhari is een manier van denken die gaat over het proces van leren en het zich meester maken van een vaardigheid. Hierin worden drie fasen onderkend op weg naar het meesterschap:

- **Shu:** leren en volgen van de regels
- **Ha:** regels weten te breken/ombuigen
- **Ri:** eigen stijl ontwikkelen

Een van de grondleggers van het Agile Manifesto, Alistair Cockburn, heeft dit concept vertaald naar het vakgebied Agile softwareontwikkeling (ref 16). Cockburn maakt de vergelijking met Agile volwassenheid. Dave Moran (ref 17) visualiseerde dit als volgt:



Shu kan beschouwd worden als het dogmatisch toepassen van de Agile practices, terwijl bij Ha en Ri de focus verschuift naar de Agile principes respectievelijk de Agile waarden. Met andere woorden, de verschuiving vindt plaats van Agile doen naar Agile denken, en daarmee in staat zijn het verschil te kunnen maken. Voor jezelf kun je besluiten of je vasthoudt aan oude/bestaande werkwijzen of dat je voor het meesterschap gaat en dus je eigen stijl ontwikkelt. Niels Talens schetste dit via een voorbeeld (ref 18):

Shu	*Volg de Scrumguide. Vertrouw dat dit raamwerk goed werkt*
Ha	*Naarmate de Agile volwassenheid toeneemt, voegen bepaalde Scrum meetings minder waarde toe (omdat ze op een andere manier ingevuld worden). De Scrum regels worden dus aangepast*

Ri	*Scrum was prima als startup, maar geldt niet meer als de enige manier van werken. Afhankelijk van de context wordt afgewogen welke Agile methode het beste bij de situatie past.*

Wil je jezelf eens toetsen op meesterschap? Er zijn diverse mogelijkheden, zoals de certificering voor Agile Practitioner vanuit het Agile Consortium (ref 19).

Niet alleen de Japanse vechtsport laat zien hoe het mogelijk is in jezelf te excelleren. Ook het Zen boeddhisme heeft tal van wijsheden die je naar een hoger niveau leiden. In elk geval vergen beide een hoge mate van concentratie en toewijding.

Ben ik nu Zen?

Of je na je lezen van dit artikel helemaal Zen bent, betwijfel ik. Maar vaak staan je alledaagse werkzaamheden op de automatische piloot en sta je niet zo gauw stil bij deze routines. Daarom is het goed je blik te verruimen om uiteindelijk je basis te verbreden en nieuwe vaardigheden te ontwikkelen. De beschreven 'wijsheden' helpen je hopelijk bij het ontdekken van betere manieren om software te ontwikkelen, te testen en anderen te inspireren. ←

Referenties

1. http://www.agilemanifesto.org
2. Pink, Daniel H. (2009), Drive: The Surprising Truth About What Motivates Us.
3. https://en.wikipedia.org/wiki/Maslow%27s_hierarchy_of_needs
4. Nonaka, Ikujiro; Takeuchi, Hirotaka (1995), The knowledge creating company: how Japanese companies create the dynamics of innovation.
5. Fægri,Tor E.(2008), Exploratory Knowledge Creation in Agile Software Development Projects. In The 31st Information Systems Research Seminar in Scandinavia.
6. https://www.scrumalliance.org/community/articles/2010/january/ba-and-knowledge-creation-in-scrum
7. Ettten, van Daan (2012), Scrum and knowledge creation, Masterthesis UvA.
8. http://www.scrumguides.org
9. Sutherland, Jeff (2010). The roots of scrum. Online presentatie (zie www.scruminc.com/roots-of-scrum-updated-accu-conference).
10. Adzic, Gojko (2011), Specification by Example. How Successful Teams Deliver the Right Software.
11. https://cucumber.io
12. http://www.specflow.org
13. http://agileleanlife.com/kaizen-growth-mindset-continuous-improvement
14. http://codekata.com
15. http://www.huibschoots.nl/wordpress/?p=493
16. http://alistair.cockburn.us/Shu+Ha+Ri
17. http://www.agileshuhari.net
18. http://blog.finalist.nl/2011/06/14/shuhari-de-weg-naar-meesterschap
19. http://www.agileconsortium.nl

DE HERKENNING VAN LEIDERSCHAP

Door Jeroen Rosink ● jeroen.rosink@gmail.com

Je kent dat wel, je bent een gedreven tester met kennis en passie voor het vak. Je weet waar je het over hebt. En als je onder gelijken bent, dan weet je meters te maken. is de situatie altijd zo ideaal? In de jaren dat ik bezig ben met het testvak heb ik het te vaak anders meegemaakt. Zo kwam ik in een praktijksituatie een keer met de teams nauwelijks vooruit. Er was of weerstand, of frustratie, maar geen succesvolle verandering. Dit gedrag begon mij op te vallen in de loop van mijn carrière.

Op zoek naar een oplossing kwam ik op een cursus Situationeel Leiderschap II gegeven door Blanchard Nederland. De theorie hiervan is gebaseerd op het model van Hershey en Blanchard.

Wat mij aantrok, is dat de aandacht van deze benadering is dat managers verandering bewerkstelligen bij hun medewerkers. Het is gericht op de mensen die je aanstuurt. En al is de aanpak op sommige vlakken formeel. Als testcoördinator of manager helpt het je succesvoller te zijn in je eigen taak of opdracht. Maar wat is situationeel leiderschap dan?

Situationeel Leiderschap

Leiderschap zoals hier bedoeld wordt, is het tonen van een gedragspatroon dat met anderen wordt gedeeld en ook zo door hen wordt gezien en ervaren.

Blanchard heeft in de loop der jaren het één dimensionale model uitgebreid naar twee dimensies. Hier onderkent men Sturend Leiderschap waarbij de aandacht ligt op het einddoel en de uitvoering van de taak. En ondersteunend leiderschap waar de onderlinge verhouding belangrijk is.

Wat mij tijdens de cursus opviel, was dat ik vooraf bij het aansturen, vertrouwde op de kennis en de skills die ik had. Maar waren die voldoende en effectief? Zelfreflectie leerde mij dat het ook anders kan. Je richt je op de taak of het doel en houdt rekening met het ontwikkelingsniveau van de persoon Het leiding geven moet hierop afgestemd worden.

Herken je leiderschap!

Probeer het zelf eens! Denk aan je eigen situatie. Gaf jij opdrachten die uitgevoerd dienden te worden om zo het testplan te kunnen volgen en te borgen? Gaf jij richting hoe jouw team het kon aanpakken? En welk resultaat bracht jou dat? Waarschijnlijk met enige moeite en soms wat gemak werd de bepaalde richting gevolgd. Maar bracht het ook de personen in jouw team verder?

Vanuit Situationeel Leiderschap II denkend kijk je per doel of taak wat het niveau is van het individu. En vanuit die gedachte stuur je de persoon aan. Of beter gezegd, afhankelijk van het niveau kies je voor sturing, coaching, ondersteuning of delegeren.

Dit klinkt logisch, maar denk even terug aan de situaties die je net hebt onderkent. Is het niet logisch dat je misschien, net als ik, voor ondersteunend gekozen hebt. Een algemene taak bijvoorbeeld: voer die test cases uit

rapporteer deze middels de gebruikelijke tool. Bevindingen leg je vast volgens de procedure. En nu wordt er getest. Heeft iedereen het begrepen? Zit iedereen op dezelfde lijn?

Mijn ervaring is dat je continu naar de signalen moet kijken, zowel van het team, maar ook het individu. Echter, daar waar Situationeel Leiderschap II handvaten geeft om het op een expliciete manier aan te pakken, is mijn ervaring dat juist bij dynamische interacties van bijvoorbeeld standups-meetings je impliciet deze skills dient te tonen. Er is geen tijd om een gestructureerd plan op te stellen. Dat neemt echter niet weg dat je de basis kunt toepassen. Dit vergt wel een extra stuk inspanning en vaardigheid.

Kortom, mijn kennismaking met Situationeel Leiderschap II heeft mij nieuwe inzichten gegeven die mij dagelijks van pas komen. ←

HET SCHAAP MET DE ACHT POTEN

Door Maarten Beks ● maarten.beks@salves.nl @MaartenBeks1

Nu het Internet of Things aan de deur klopt, lijkt het credo van technologiebedrijven meer dan ooit 'snel, sneller, lichtsnelheid'. De strijd tussen time-to-market en kwaliteit wordt daarmee nog heftiger. Om als tester deze strijd in het voordeel van kwaliteit te beslechten, moeten we de juiste vaardigheden bezitten. Maar welke? En welke vaardigheden zijn specifiek voor testautomatisering vereist?

Vijf poten niet voldoende

Het spreekwoordelijke testschaap met de vijf poten struikelt tegenwoordig over zijn eigen ledematen. Een 'moderne' tester moet zeven of soms wel acht poten hebben om de hooggespannen verwachtingen op het gebied van kwaliteit waar te maken. Wat moet een tester allemaal kunnen?

- een uitgebalanceerde teststrategie opstellen;
- een teststrategie overtuigend verkopen aan de stakeholders;
- de acceptatietest coördineren;
- met testspecificatietechnieken testgevallen opstellen;
- testgevallen automatiseren en bij voorkeur ook embedden in een continuous integration-oplossing;
- optreden als scrummaster;
- gedetailleerde kennis hebben van een veelvoud aan domeinen zoals zorg, de publieke sector, de financiële wereld en de industrie;
- tegelijkertijd uitvoeren van een usability- en een performancetest.

Vier pijlers

Laten we eens even generaliseren. De kunde van een tester rust op vier pijlers: kennisniveau van 'het testvak', generieke ICT-kennis, domeinkennis en soft skills. Sommige collega's onderscheiden zich als uitmuntend testanalist, terwijl anderen met domeinkennis van grote toegevoegde waarde zijn voor een project. Om een succesvol tester te zijn, is het essentieel dat de kennis en kunde van een tester voor elke pijler van voldoende niveau is (waarbij we vooral het belang van domeinkennis en soft skills niet moeten onderschatten!).

Stabiele testautomatisering

Terug naar de strijd tussen time-to-market en kwaliteit. Hoe kunnen we kwaliteit waarborgen in een zo snel veranderende wereld? De vraag naar een steeds kortere time-to-market met behoud van kwaliteit kan alleen kostenneutraal beantwoord worden door de inzet van kwalitatief hoogstaande, stabiele testautomatisering.

Een ogenschijnlijk simpele oplossing, maar het succesvol implementeren van testautomatisering is geen sinecure. User interfaces gedragen zich immers zelden stabiel, en herhaalbaarheid van testgevallen is in verband met controle over testdata al vaak uitgeroepen tot 'de hel op aarde'. Bovendien is er een spaghetti aan devices, operating systems en browsers waarop functionaliteit ontsloten moet kunnen worden. Tot slot kunnen we vanwege haperende beheerprocessen rondom de applicatie-lifecycle nauwelijks voorspellen wat zich allemaal op een testomgeving afspeelt.

Randvoorwaarden

Stabiliteit en controle zullen steeds belangrijker worden voor testers. De testautomatiseerder van de toekomst zal zich - om testautomatisering succesvol te kunnen implementeren - daarom ook om de randvoorwaarden moeten bekommeren. Dus niet alleen de tool bedienen, maar ook zorgdragen voor:

- implementatie van beheerprocessen rondom de applicatie-lifecycle;
- afdwingen van controle over testdata;
- maken van een op risico en stabiliteit gebaseerde keuze in de devices, operating systems en browsers waarop een test gedraaid moet worden;
- definiëren van een testautomatiseringsstrategie die erop is gericht zo veel mogelijk testen geautomatiseerd uit te voeren zonder gebruik te maken van een GUI.

Wacht, willen we soms zelf het schaap met de acht poten zijn? Toch niet. Doordat Agile al gemeengoed is, zijn grenzen tussen de verschillende rollen al vervaagd en neemt de vraag naar multidisciplinariteit toe. Deze ontwikkeling zal de komende tijd verder doorzetten. Een soort single point of dev ops voor testers.

Beperk testen via de GUI

De testautomatiseringsstrategie die erop is gericht zo veel mogelijk testen uit te voeren zonder gebruik te maken van de GUI, verdient wellicht nog wat extra toelichting. De redenering hierachter is dat een GUI er enkel en alleen is om functionaliteit te ontsluiten. Een GUI op zich heeft geen bestaansrecht en is vaak ook nog eens de meest instabiele component in de keten. Definieer daarom een op risico gebaseerde testautomatiseringsstrategie die erop is gericht is om zo veel mogelijk geautomatiseerd te testen via unit-testen en de API's die de functionaliteit ontsluiten. Beperk geautomatiseerde testen via de GUI en ga efficiënt met tijd en middelen om. Waarom zou je immers in een systeemtest iets testen als je dit binnen een unit-test al aantoonbaar met de juiste dekking hebt getest?

Dit heeft uiteraard zijn weerslag op de rol van de tester. Hebben we ook nog kennis nodig van unit-testen. Oeps, nog een extra poot voor het schaap… ←

TEST DE ACCEPTANT

Door René Lak ● Rene.Lak@bartosz.nl @Testlakkie

Testers hebben te maken met een acceptant. Deze acceptant bepaalt of het werk voldoende vertrouwen geeft om in productie te gaan met het nieuwe of aangepaste systeem. Hoe vaak komt het niet voor dat deze acceptant nog aanvullende testen uitgevoerd wil zien? Of ergerlijker: helemaal geen tijd heeft voor de tester? De acceptant heeft te maken met testers. Deze testers moeten het vertrouwen geven dat het goed zit met de nieuwe functionaliteit. Vaak ontstaat het idee dat de tester zich niet bewust is van wat er in productie met die functionaliteit moet gebeuren. Of vervelender nog: de tester is niet te volgen en rapporteert onduidelijk. In dit samenspel draait het om verwachtingen. Hoe kunnen we daarmee omgaan?

De tester

De tester is een ingehuurde specialist, niet een super-user die ook het testwerk erbij doet. Het maakt voor de tester niet uit in welke business er getest moet worden. De manier van denken en de aanpak is hetzelfde.

Onze tester maakt een testplan en –strategie en legt deze voor aan de acceptant. Met wat kleine op- en aanmerkingen kan de tester daarna aan de slag gaan. Agile of waterval, dat maakt in principe niet uit. In de Agile situatie worden onduidelijkheden alleen iets sneller besproken.

Voor de functionele testen zal hij op basis van de specificaties testgevallen creëren en vervolgens uitvoeren. Als de gevonden bugs zijn opgelost en alle testen binnen de gestelde tijd afgerond zijn, hoeft hij alleen de testresultaten nog te laten accepteren. Daartoe stuurt hij de gebruikte Excel-sheet met testgevallen en resultaten in een e-mail naar de acceptant. Of geeft de resultaten in een demo weer.

Wat mag de tester in dit geval verwachten van de acceptant? Waarschijnlijk niet veel. Als de tester vertrouwen geniet, dan zal er mogelijk na een korte blik van de acceptant een akkoord volgen. Is dat niet het geval, dan kan het wel eens zo zijn dat onze tester een antwoord krijgt als ‘Hier kan ik niets mee’. Of ‘Van Pietje krijg ik het altijd in een ander formaat’. Stopt de acceptant er wél een hoop werk in, dan volgen er vragen zoals: ‘Waarom is dit wel getest en waarom dat niet?’ ‘Wat wordt bedoeld met “Passed”?’, ‘Wat is er dan precies gedaan?’ of ‘Is dit zomaar te vertrouwen?’

Het acceptatieproces vanuit de tester

Hoe is het acceptatieproces te stroomlijnen? Voor de tester begint het acceptatieproces al direct aan het begin van het project. Daarom is het goed om:

1. De acceptant mee te nemen in de aanpak. Maak hem of haar er deelgenoot van. Niet door een enkele review van het testplan maar door actief te overleggen over de aanpak.
2. Kritisch te zijn met betrekking tot de noodzaak van bepaalde gevraagde testsoorten. Zorg dat je kan uitleggen waarom deze testen niet nodig zijn.
3. De testgevallen te bespreken en aan te geven welke risico’s deze afdekken.
4. Te vragen of de acceptant voorbeelden vanuit de praktijk heeft. Deze kunnen wellicht eenvoudig omgezet worden in testgevallen. En belangrijker, de acceptant voelt zich gehoord.
5. Van tevoren af te spreken in welk formaat er gerapporteerd wordt en met welke output.

6. Afspraken te maken over met welke regelmaat wordt gerapporteerd. Overvoeren is niet aan te raden. Maar alleen aan het einde rapporteren ook niet. Zeker niet als het project uitloopt, want als er dan nog extra testen gedaan moeten worden, dan moeten er goede argumenten aangeleverd worden waarom er nog meer tijd nodig is en waarom je dat niet had kunnen voorzien.
7. Deze afspraken in het team als standaard te gebruiken, zodat niet iedere tester opnieuw het wiel uitvindt of op een eigen manier rapporteert.

En wil de acceptant geen tijd vrijmaken? Wijs hem dan op het gevaar van een werkend systeem dat niet doet wat de acceptant verwacht.

De acceptant

De acceptant is hier de business specialist; de end user en/of de producteigenaar. Deze is dagelijks met de materie bezig en wordt van alle kanten bestookt met vragen en problemen. De tester is slechts één van de velen. Het maakt voor de acceptant niet uit hoe er getest wordt, als het maar vlekkeloos werkt in productie.

Onze acceptant ontvangt een testplan. Het blijkt een 'cover your ass' verhaal en de acceptant begrijpt niet waarom er niet beschreven wordt wat er echt nodig is. De business termen die er in staan kloppen niet en de acceptant vraagt zich af wat het moet worden.

Dan volgen de testresultaten. 'Hoe moeten die nu weer gelezen worden?', of 'Dat is heel anders dan wat de collega tester gisteren aanleverde.' Ook wel gehoord: 'Geen idee wat hier bewezen wordt', en 'Niet akkoord.'

Wat mag de acceptant in dit geval verwachten van de tester? Waarschijnlijk ook niet veel. De tester is op basis van documentatie gaan testen. De acceptant had geen tijd voor weer hetzelfde verhaal. Testen gebeurt al zo veel dat de testers moeten weten wat er verwacht wordt. Toch is het belangrijk voor de tester om te weten hoe de wijziging in het grote plaatje past en waar de risico's zitten. Anders volgt er mogelijk een prima werkend systeem, maar dat niet doet wat er is gevraagd.

Het Acceptatieproces vanuit de acceptant

Vaak is de acceptant al betrokken bij de eerste ideeën voor het nieuwe/gewijzigde systeem. Alles zit al in het hoofd. Het acceptatieproces is te stroomlijnen door:

1. De tijd te nemen om het hoe en waarom toe te lichten wanneer de tester bij het project komt.
2. Voorbeelden te geven waarom de aanpassingen in het systeem gemaakt moeten worden. Deze kan de tester wellicht gebruiken voor de test. En belangrijker: zowel de tester als de acceptant hebben hetzelfde beeld.
3. Samen met de tester de testscripts door te lopen, zodat je weet wat er getest gaat worden. Als er dan 'Wat als' vragen oppoppen, dan mist er wellicht een testgeval.
4. Goed af te stemmen wat wanneer verwacht wordt. De acceptant weet hoeveel tijd beschikbaar is voor de acceptatie.
5. Van tevoren af te stemmen hoe de rapportage aangeleverd moet worden. Dat scheelt een hoop ergernis.
6. De tester aan de afspraken houden, ook al loopt het project uit. Houd geregeld contact zodat bekend is hoe de testzaken er voor staan. En dus ook het uiteindelijke product.

Test de acceptant?

Soms heeft de acceptant het gevoel dat hij tester moet zijn. Of dat hij getest wordt op zijn analytisch vermogen. Soms heeft de tester het gevoel dat hij moet raden wat de criteria van de acceptant zijn. Dat is niet nodig. Meer begrip voor de ander en elkaars werk zorgt voor een vlottere acceptatie. Dit kan door beter verwachtings-management van beide kanten. Beide partijen kunnen elkaar het leven makkelijk maken door goede afspraken en elkaar daar ook aan te houden. ←

TESTNET NIEUWS

TestNet Nieuws is een uitgave van de Vereniging TestNet, een bloeiende vereniging met meer dan 1700 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