AGILE TESTEN

Testen als teamsport: het antwoord op innovatie in ontwikkelprocessen

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Tijman, A.

Agile Testen. Testen als teamsport: het antwoord op innovatie in ontwikkelprocessen /
A. Tijman. – Nieuwegein: Ordina
ISBN: 978-90-78244-02-8
NUR 980
Trefw.: informatica; testen

Ontwerp en omslag: vandenberg drukwerken, Maarn

INHOUDSOPGAVE

VOORWOORD

"Testing is an extremely creative and intellectually challenging task" schreef de grondlegger van gestructureerd testen, Gelford Myers. Met dit testprincipe in het achterhoofd kun je de ontwikkelingen rond agile software development en de rol van testen het beste samenvatten. Testen wordt niet anders, maar agile testen vereist wel meer van de testers!

De tijd dat de diverse agile ontwikkelmethoden (XP, RUP, DSDM, Scrum, etc.) konden worden afgedaan als een hype ligt inmiddels achter ons. Steeds meer organisaties stappen over van de traditionele watervalmethode naar een agile ontwikkelmethode, en veelal met succes. In mijn eigen praktijk word ik steeds vaker geconfronteerd met organisaties die agile practices daadwerkelijk toepassen in hun projecten, zoals iteratief ontwikkelen, timeboxing, stand-up meetings, continuous integration, test driven development en vroegtijdige validatie. Ik behoor tot de groepering 'eerst zien dan geloven', maar inmiddels ben ik ervan overtuigd dat er veel organisaties zijn waar de diverse agile ontwikkelmethoden goed kunnen worden toepast. Dit geldt met name indien sprake is van een hoge mate van dynamiek in de business.

In essentie is de discussie rondom agile terug te voeren naar de kwaliteitsgoeroes van weleer. Crosby met zijn klassieker "Quality is Free" gaat uit van de maakbare wereld. Daarin worden eerst de requirements volledig uitgedetailleerd alvorens wordt begonnen met ontwikkeling en productie. Dat was in feite de allereerste versie van het watervalmodel. Bij Crosby heeft een systeem kwaliteit als het voldoet aan de vastgestelde requirements. Juran daarentegen stelt het begrip 'fitness-for-use' centraal. Bij hem is sprake van kwaliteit als er een hoge mate van klant- c.q. gebruikerstevredenheid is. Agile methoden volgen feitelijk de filosofie van Juran.

Wat betekent dit nu voor de tester – kan alle huidige TMap® kennis als verloren worden beschouwd? Integendeel! Testers dienen hun testmethoden en -technieken nog veel beter te kennen. Ze dienen niet alleen te weten hoe een en ander moet worden toegepast, ze dienen ook te begrijpen wat de achtergronden en doelstellingen zijn. Feitelijk wordt van de tester gevraagd om de test practices uit TMap® en het internationale testopleidingsprogramma ISTQB te kunnen toepassen binnen een andere context. Dit stelt hoge eisen aan het niveau van de tester. Begrijpen waaróm je iets

doet is iets anders dan het kunnen invullen van een template. Niet alleen aan de testkennis en -kunde worden hoge eisen gesteld, het nauw samenwerken in een team stelt ook hoge eisen aan de communicatieve vaardigheden van de testprofessional.

In dit boek behandelt Anko Tijman het begrip agile en de gewijzigde rol van gestructureerd testen. Hij gaat in dit boek niet zozeer in op de concrete hoe en wat-vragen, maar veel meer op de vraag: waarom? Op deze wijze maakt de lezer kennis met het begrip agile en leert hij hoe hij zijn huidige bagage aan testkunde kan toepassen in de agile omgeving. Het testplan en de testgevallen blijven immers relevant, maar hebben een andere, minder omvangrijke vorm. Recentelijk werd ik geconfronteerd met een agile testplan van slechts één pagina, bestaande uit de alom bekende, op requirements gebaseerde risico-matrix en een beschrijving van een gedifferentieerde testaanpak per kwadrant. Dit bleek in de betreffende (agile) omgeving voldoende om aan de doelstellingen van het testplan te voldoen!

Anko Tijman behoort tot de groep van early adopters op het domein van testen in agile projecten. Zijn kennis is gebundeld in dit boek, voorzien van een groot aantal praktijkvoorbeelden. Als je meer wilt weten van agile ontwikkelen en de rol van testen, start hier.

Drs. Erik van Veenendaal, CISA
Directeur Improve Quality Services BV
Dommelen, december 2006

INLEIDING

Als er één vakgebied is waar de veranderingen niet op zich laten wachten, is het de ICT wel. Veranderingen worden hypes, hypes worden trends, en trends worden ontwikkelingen. En uiteindelijk worden die ontwikkelingen geaccepteerd als gemeengoed.

Dit boek beoogt vanuit het perspectief van het software testen een antwoord te geven op een ontwikkeling die zich al langer afspeelt: het ontwikkelen van software volgens 'agile' principes. Deze principes omvatten een iteratief, incrementeel ontwikkelproces, dat een zo hoog mogelijke waarde voor de business wil leveren. Een IT-ontwikkeling die waarde voor de business oplevert – dat heeft u vaker gehoord. Wat is vernieuwend aan agile softwareontwikkeling? Het kan zich daadwerkelijk aanpassen aan wijzigende wensen en voortschrijdend inzicht. Agile ontwikkelen wíl kunnen veranderen en koste wat het kost een werkende oplossing creëren waar de business waarde aan toekent. Voor de business biedt agile ontwikkelen geweldige mogelijkheden – maar om dit mogelijk te maken moet er wel wat in de ICT veranderen. Het veranderdilemma waar de ICT voor staat, wordt beschreven in hoofdstuk 1.

Vanuit een traditioneel testersperspectief is de ontwikkeling naar een flexibeler ontwikkelproces zorgwekkend: van oorsprong geldt het software testen als een discipline waar veranderingen in requirements op weerstand stuiten en waar het releasen van software vaak liever wordt uitgesteld omdat de kwaliteit nog niet voldoende is. Echter, agile softwareontwikkeling is een aanpak waarin testen een veel belangrijker rol speelt dan veel testers denken. Het biedt legio mogelijkheden, een groot aantal daarvan zullen in dit boek geschetst worden. Agile ontwikkelen staat een integrale aanpak van ontwikkelen voor. Alle noodzakelijke activiteiten worden in één stap, per iteratie doorlopen. Het testen is hier een volledig geïntegreerde stap in. Agile ontwikkelen kan niet zonder een geïntegreerd testproces. Het testen biedt een van de belangrijkste waarden voor een agile team, namelijk feedback. De nieuwe normen en waarden voor testen in een agile context worden beschreven in hoofdstuk 2, ´De Agile Testen Aanpak´.

Wanneer we fundamenteel anders met software-ontwikkeling omgaan, betekent dit nogal wat voor onze huidige werkwijzen en principes. In hoofdstuk 3 ´De Impact van Agile´ wordt uitgelegd voor welke uitdagingen medewerkers en IT-organisaties staan. Vanuit businessperspectief wordt gekeken welke mogelijkheden agile ontwikkelen biedt om strategische keuzes te kunnen ondersteunen.

Wanneer er al in uw organisatie volgens agile principes software ontwikkeld wordt, dan zal dit boek richting kunnen geven aan een gestructureerde aanpak van het software testen. Wanneer die beslissing nog niet is genomen maar wel overwogen wordt, dan kan dit boek leidend zijn voor een inbedding van het testen in de agile projecten.

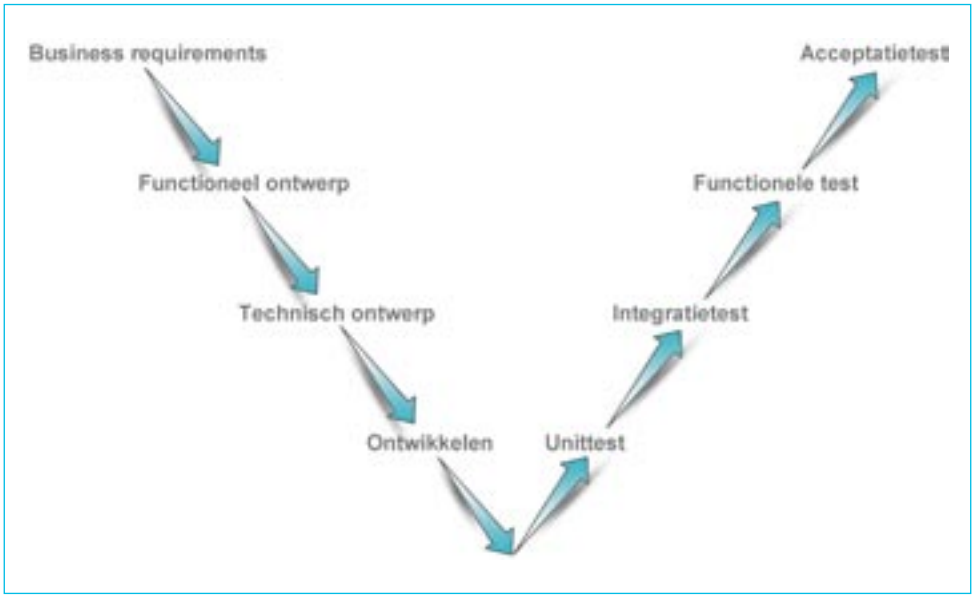
1. HET DILEMMA IN SOFTWARE TESTEN

Het software testen in Nederland bestaat nu circa vijftien jaar. Jaren waarin veel werk verzet is, zoals het uitbrengen van boeken, het ontwikkelen van methodes en het bewerkstelligen van certificatie van het softwaretestvak. Al deze bewegingen hebben tot doel gehad om het testen een professioneel aanzien te geven in de dynamische wereld van de softwareontwikkeling. Petje af voor diegenen die hier allemaal een zinvolle bijdrage aan hebben geleverd, en zo ook het testen in Nederland op de kaart hebben gezet. Ten opzichte van andere Europese landen heeft Nederland een sterke testcultuur ontwikkeld. Al begin jaren negentig verschenen de eerste artikelen van Nederlandse makelij, met als hoogtepunt de publicatie van de testmethode TMap®[1].

Op dit moment is TMap® dé standaard aanpak voor gestructureerd testen in Nederland. Juist nu de methode een dergelijke status heeft verworven, blijkt dat de business eisen stelt die hier haaks op staan. Er is vraag naar meer flexibiliteit, transparantie en participatie in het testproces. Dit dilemma wordt in dit hoofdstuk uitgewerkt.

1.1. TRADITIONELE SOFTWAREONTWIKKELING

Vrijwel alle initiatieven voor het professionaliseren van testen, en TMap® in het bijzonder, sloten aan bij de ontwikkelmethode die op dat moment gemeengoed was, de zogenoemde watervalmethode.

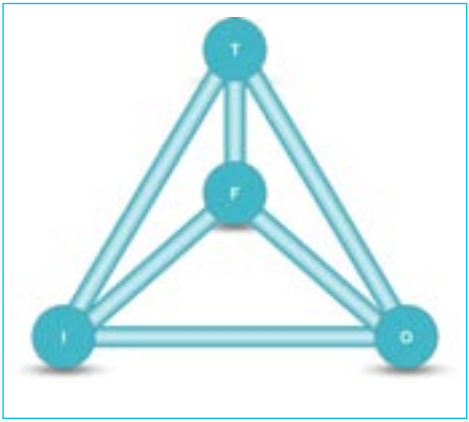


Afbeelding 1-1. Het V-model waarin ieder stadium één keer wordt doorlopen.

Deze methode kent diverse benamingen en methodevarianten: Watervalmethode, LAD (Lineair Application Development), V-model (model met expliciete testactiviteiten), SDM / SDM-II (Systems Development Method). Wat deze termen met elkaar delen is een aanpak gekenmerkt door:

- Een gescheiden fasering van de verschillende stadia van het ontwikkelproces (Informatieanalyse, Functioneel Ontwerp, Technisch Ontwerp, Bouwen, Testen en Implementatie).
- Een aanvang van het volgende ontwikkelstadium op het moment dat het voorgaande volledig is uitgevoerd en aan de exit-criteria voldoet.
- Het borgen van de kwaliteit van tussenproducten door middel van reviews.
- Het testen van (tussen-)producten achteraf.
- De inzet van specialisten voor specifieke stadia: ontwerpers voor het ontwerpstadium, ontwikkelaars voor het ontwikkelstadium en testers voor het testen.
- Het toekennen van specifieke verantwoordelijkheden aan elke specialisatie.
- De ondersteuning door een planmatige, taakgebaseerde projectmanagementaanpak

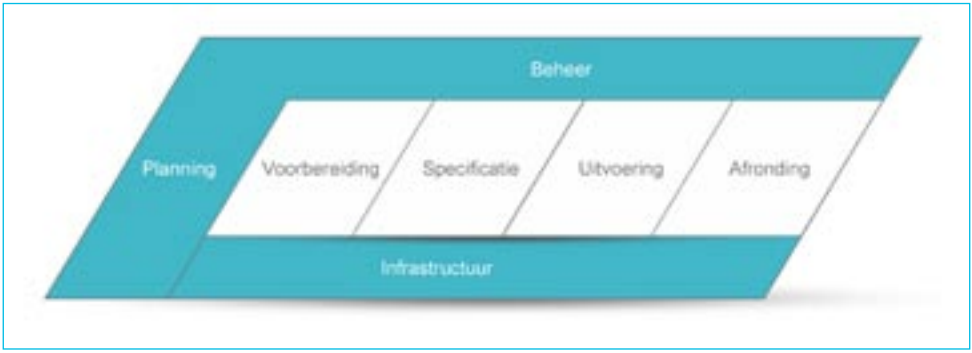
De leidende gedachte achter dit één-stadium-één-fase model is: een activiteit eenmalig uitvoeren is efficiënter. Deze gedachte is inmiddels ingehaald door de werkelijkheid. Agile ontwikkelen is niet gefaseerd, maar integraal ontwikkelen. Dit leidt sneller tot werkende producten die tastbaarder zijn voor de business. Deze vorm van feedback maakt het proces als geheel vele malen effectiever.



Afbeelding 1-2: De vier pijlers volgens TMap®

Het watervalmodel is de basis van de testmethode TMap®, op dit moment de *de facto* standaard van testen in Nederland. TMap® kent vier pijlers:

- Fasering: het testproces valt uiteen in de fases planning en beheer, voorbereiding, specificatie, uitvoering en afronding.
- Technieken: gedefinieerde werkwijzen om tot betere testgevallen te komen en een hoge dekkingsgraad van de test te realiseren.
- Organisatie: inrichtingsaspecten van de organisatie om afgestemd te zijn op en rekening te houden met het testproces, zoals beheertaken, testrollen en testfuncties.
- Infrastructuur: de eisen aan een testomgeving, het gebruik van testtools en kantoorinrichting.



Afbeelding 1-3: De fasering van het testproces volgens TMap Next®[2]

Vanuit die watervalachtergrond kan TMap® goed omgaan met projecten waarin eenduidige requirements zijn gevat in een 'bevroren' functioneel ontwerp, waarin er voldoende tijd is ingeruimd voor het voorbereiden en uitvoeren van testscripts, en waarin het systeem volledig uitontwikkeld en stabiel wordt opgeleverd ter test. Niks mis mee. In de praktijk blijkt het verschil tussen theorie en praktijk groter dan de theorie doet vermoeden.

De strikte fasering van de watervalmethode werkt onder de volgende voorwaarden:

- Besluiten die worden genomen zijn goed. Omdat de cyclus maar één keer wordt doorlopen, worden besluiten maar één keer genomen. Daardoor zijn ze achteraf ook moeilijk wijzigbaar, omdat dan de hele cyclus opnieuw moet worden doorlopen.
- De klant weet van te voren exact wat hij wil: wat zijn huidige proces is en hoe het systeem er uit moet gaan zien. Voortschrijdend inzicht kan slechts toegepast worden door middel van wijzigingsvoorstellen, waarvoor een apart (financieel) traject is opgesteld.
- De reviews die de kwaliteit van de tussenproducten moeten waarborgen, moeten ook daadwerkelijk uitgevoerd worden en leiden tot de noodzakelijke veranderingen.
- Aan het eind van iedere fase is het stadium ook echt afgerond, dat wil zeggen dat de organisatie in de volgende fase direct aan de slag kan met hetgeen is opgeleverd.

In deze context sluit TMap® als testmethode naadloos aan op het project en is een agile aanpak niet zinvol.

1.2. RESULTATEN VAN WATER-VALPROJECTEN

Ruwe schattingen laten zien dat slechts dertig tot vijfendertig procent van de projecten waar de watervalmethode wordt gehanteerd als succesvol mogen worden beschouwd – binnen tijd en budget, en voldoende getest. De vraag is dan ook: voldoet deze strikte aanpak in een wereld die sneller nieuwe eisen stelt aan de business dan voorheen, waarbij de techniek steeds complexer wordt?

Projecten die niet aan de watervalcriteria voldoen, worden vaak geschaard onder de noemer ad hoc* projecten. Deze projecten kenmerken zich door:

- continu wijzigende specificaties
- tegenvallende progressie van het ontwikkelteam
- krimpende testtijd
- de te testen software voldoet zelden aan de gestelde entry-criteria
- matige acceptatie door eindgebruikers
- een discussie rondom de opgeleverde waarde van de software

Dit zijn over het algemeen de vijfenzestig tot zeventig procent van de projecten die als niet-succesvol de boeken ingaan.

* Ad Hoc (Lat): voor een specifiek doel (en dus niet, zoals het vaak gebruikt wordt, ongestructureerd)

Kijken we naar gestructureerd testen in de traditionele vorm, dan zien we een soortgelijke afhankelijkheid van expliciete en impliciete aannames:

- Het testproces dient gefaseerd te verlopen, dus gescheiden in stadia en activiteiten.
- Specificaties dienen juist, volledig en consistent te zijn om de testtechnieken te kunnen toepassen.
- Het resultaat uit de ontwerp- en ontwikkelfasen dient aan de gestelde exit-criteria te voldoen.

"If knowledge can create problems, it is not through ignorance that we can solve them." – Isaac Asimov

Testen gebeurt aan het eind van het project. Vaak is er dan al vertraging opgelopen en staat het testen onder druk. Vervolgens moet er afgeweken worden van de teststrategie en blijven er meer fouten dan strikt noodzakelijk in het systeem zitten. Het is in feite net zoals Cem Kaner (professor Software Engineering aan The Florida Institute of Technology) al schreef in 1988 in zijn boek 'Testing Computer Software'[3]: *"Some books say that if our projects are not 'properly' controlled, if our written specifications are not always complete and up to date, if our code is not properly organized according to whatever methodology is fashionable, then, well, they should be. These books talk about testing when everyone else plays 'by the rules.' This book is about doing testing when your co-workers don't, won't and don't have to follow the rules."*

Als veel projecten er niet in slagen om de watervalmethode succesvol uit te voeren, voldoet het dan wel aan de eisen die we stellen aan een methode? Wellicht is het model gewoon bijzonder moeilijk passend te krijgen op de praktijk van vandaag de dag.

1.3. HEDENDAAGSE ONTWIKKELPROCESSEN

In de afgelopen jaren ontstonden nieuwe ontwikkelprocessen. Dat begon bij RAD (Rapid Application Development, JAD (Joint Application Development) en IAD (Iterative Application Development), dat min of meer gestalte heeft gekregen in DSDM (Dynamic Systems Development Method). Al in 1994 werd deze ontwikkelmethode geformaliseerd door de oprichting van het DSDM consortium. Een beweging eind jaren negentig vanuit de Verenigde Staten resulteerde onder meer in eXtreme Programming (XP), Scrum en Lean Software Development, met zijstromingen als Agile Modeling, Crystal en Feature Driven Development. In 2001 verzamelden de vertegenwoordigers van deze methodieken zich, met als resultaat het Agile Manifesto*:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions
over processes and tools
Working software *over*
comprehensive documentation
Customer collaboration
over contract negotiation
Responding to change
over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

(in dit kader zijn de 'items on the left' vetgedrukt)

* www.agilemanifesto.org

Hier wordt duidelijk stelling genomen voor een ander soort software ontwikkeling. Agilisten prefereren nieuwe normen en waarden boven de traditionele normen en waarden. Dat betekent niet dat ze die 'oude' waarden niet belangrijk vinden, maar dat ze andere succesfactoren definiëren. De succesfactor van een project wordt vaak gevormd door de mensen en hun communicatie. Of door de samenwerking met de klant, door de uiteindelijke kwalitatieve werking van de software, of het feit dat de projectorganisatie in staat was te kunnen omgaan met veranderingen. De aanpak die dit beoogt noemen we 'agile'. Letterlijk vertaald wil dat zeggen: wendbaar, flexibel. Een flexibele aanpak voor softwareontwikkeling dus, die zich richt op de volgende aspecten:

- Hoge business value leveren, door nauwe en actieve klantbetrokkenheid.
- Uitgaan van veranderingen gedurende het project.
- Frequent kwalitatief goede, werkende software opleveren.
- Frequent evalueren van geldende werkwijzen met als doel zo effectief mogelijk te zijn.

De impact van agile ontwikkelen op de organisatie en de medewerkers wordt verder uitgewerkt in hoofdstuk 3.

Volgens een onderzoek van Forrester Research uit november 2005[4] blijkt dat op dat moment veertien procent van de Amerikaanse en Noord-Europese IT-bedrijven aan agile ontwikkeling doen. En dat negentien procent van de overige bedrijven in agile softwareontwikkeling geïnteresseerd is of van plan is het te gaan doen. Een beweging om serieus rekening mee te houden. We moeten dus op zoek naar een testmethode die hierop aansluit.

1.4. EISEN AAN AGILE TESTEN

Als we kijken naar bovengenoemde doelen van agile methodes, wat zijn de eisen die we moeten stellen aan een agile testaanpak? Deze aanpak moet:

- communicatie met de klant en met het team faciliteren en aanmoedigen
- flexibel zijn om mee te kunnen bewegen met de projectveranderingen
- in staat zijn om vaak en veel te testen
- voortschrijdend inzicht makkelijk kunnen toepassen
- eenvoudig te begrijpen zijn
- eenvoudig toe te passen zijn door alle teamleden

Kortom, deze aanpak brengt nogal wat veranderingen met zich mee. Gelukkig leert de ervaring dat testen ook gewoon testen blijft, en de mate van aanpassing zich met name op het gebied van attitude en verwachtingen bevindt. Hetzelfde werk – aantonen van de kwaliteit van het product – met een andere instelling. Hierover meer in het volgende hoofdstuk.

Hoe zeer de watervalmethode ook de standaard is geworden voor softwareontwikkeling, lang niet altijd levert het de resultaten op die het beoogt. Omdat het traditionele testen zich vooral gericht heeft op deze aanpak, is het op essentiële punten niet geschikt voor toepassing in een agile context. Het is tijd om serieus na te denken over onze huidige testconventies. De watervalmethode biedt een gestructureerde en geaccepteerde methode voor testen - maar we moeten het ook kunnen loslaten om tegemoet te komen aan de veranderende eisen die de business stelt aan ontwikkeltrajecten.

2. DE AGILE TESTEN AANPAK

In dit hoofdstuk schetsen we de contouren van de agile testen aanpak. Wat de normen en waarden van een agile project zijn, welke normen en waarden we onderscheiden voor een agile testproces, wat de verschillen tussen traditioneel en agile testen zijn en wat dit betekent voor het testen in de praktijk.

2.1. NORMEN EN WAARDEN

Agile ontwikkelen is een mindset waarin er andere normen en waarden gelden, waar andere doelstellingen voor software ontwikkelen worden gesteld. Willen we met een testaanpak hierop aansluiten, dan zullen we als testers goed naar onze eigen normen en waarden moeten kijken. Normen en waarden zijn de sturingsinstrumenten voor prioriteiten die je in je project stelt en de activiteiten die je uitvoert. Het is datgene waarnaar je verwijst als je gedwongen wordt om keuzes te maken. Dat zijn de zaken waaruit je je meerwaarde levert. Meestal zijn het generieke begrippen, soms universele waarden waarover veel mensen het eens zijn. Maar dat wil niet zeggen dat het makkelijk is om dat dag in dag uit in een project te verwezenlijken.

Een van de boeken die aan de wieg heeft gestaan van de agile beweging in de Verenigde Staten is 'Extreme Programming Explained' van Kent Beck uit 1999[5]. Daarin geeft de auteur nadrukkelijk uitleg over nieuwe normen en waarden voor softwareontwikkeling. Initieel begon hij met een viertal normen en waarden ('values'), te weten:

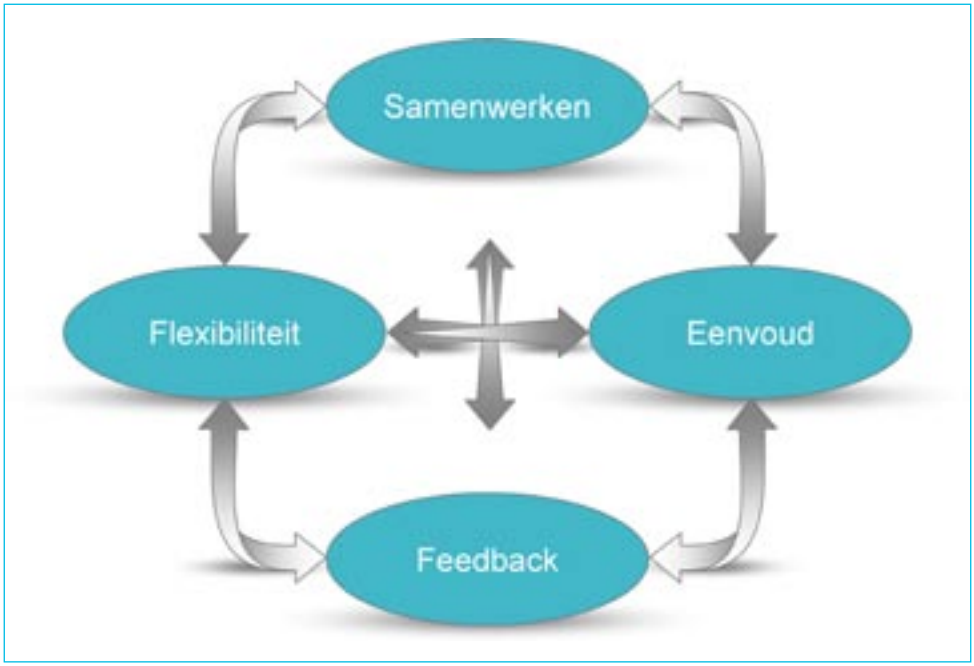
- Communicatie
- Feedback
- Eenvoud
- Moed

Inmiddels heeft hij er daar een aantal aan toegevoegd[6] en rekent hij ook universele waarden tot de essentie van eXtreme Programming (XP), zoals:

- Respect
- Integriteit
- Eerlijkheid
- Verantwoordelijkheid

De methode XP* is voornamelijk gebaseerd op ontwikkelpractices, zoals het gebruik van unit testen en het continu integreren van code en het refactoren van code. Het boek gaat niet diepgaand in op testen – eigenlijk beschrijft geen van de agile methodes het testen in detail. Dus hebben we de handschoen maar opgepakt: we hebben door een testbril gekeken naar agile waarden. En vervolgens onze ervaringen met traditioneel testen en het testen in agile projecten op één hoop gegooid en alles samengevat tot de kern.

De agile normen en waarden die wij karakteristiek vinden voor het toepassen van een effectief en efficiënt testproces in een agile project zijn: samenwerken, flexibiliteit, eenvoud en feedback. Met deze nieuwe waarden op het gebied van testen zijn we beter in staat om de wens van de klant te volgen. Daarnaast kunnen we met het testen sneller en beter inzicht verschaffen in de waarde die de nieuwe software creëert voor de business.



Afbeelding 2-1: De vier waarden in agile software testen

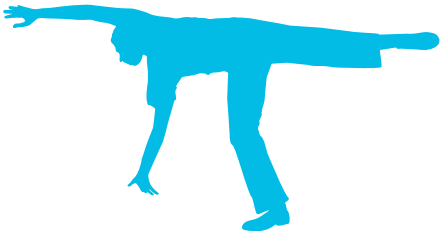
* www.xprogramming.com

2.1.1. SAMENWERKEN

Net zoals voor ieder ICT-project geldt, dient er in een agile project op hoog niveau een goede samenwerking te zijn tussen opdrachtgever en opdrachtnemer. Maar ook in het projectteam zelf dient er nauw samengewerkt te worden. Sterker nog: er moet op gestuurd worden. Maar hoe vaak gebeurt dat? Moeten testers niet onafhankelijk zijn? Objectief? Hebben ze hun eigen verantwoordelijkheid? Ja en nee. Onafhankelijkheid of objectiviteit ligt niet in het feit of de tester op een andere kamer zit. Zet testers dus bij het projectteam op de kamer om ze beter te laten samenwerken. Als de kamer daarvoor niet groot genoeg is, stel dan multifunctionele teams samen. Kwaliteit is in agile projecten een teamverantwoordelijkheid, en dus niet iets van de testers alleen. Kwaliteitsproblemen zijn zaken die het hele projectteam aangaan. Er moet veel overlegd en samengewerkt worden. En dan moet je dicht bij elkaar zitten. Agile ontwikkelen is een teamsport.

Sturen op samenwerking

Ik kwam nieuw in een projectteam waar alle zeven projectleden op één kamer zaten. De randvoorwaarden voor een team dat goed samenwerkt waren aanwezig: whiteboards, flipovers, geen rommel, voldoende licht en ruimte. Toch functioneerde het team niet. De assistent-projectleider kwam op een ochtend binnen met een probleemmelding, legde het uit en voegde er aan toe: "Ik verwacht van jullie dat jullie goed samenwerken" – en vertrok. Hij liet mij verbaasd achter: waar was de sturing hierop gebleven? Alleen de verwachting uitspreken en faciliteren is niet voldoende.



"None of us is smart as all of us" – Gerald Weinberg

Waar we verder dieper op ingaan is het feit dat samenwerking voor een deel ook je testbasis is. Agile projecten gaan uit van ruwe requirements, dat wil zeggen dat van te voren nog niet in detail vaststaat hoe de requirements eruit gaan zien. Wel is van te voren duidelijk dat de requirement zinvol is: een bepaalde waarde vertegenwoordigt, en belangrijk is. Een agile projectteam realiseert zich namelijk dat het ontzettend lastig is om details in één keer goed te voorspellen. Dan kun je beter de details laten sturen door het proces zelf: interactie tussen klant, ontwerper, ontwikkelaar en tester die ieder hun expertise gebruiken om het wél in een keer goed te doen.

Een belangrijk aspect daarbij is wel dat de noodzakelijke expertise in de samenwerking beschikbaar moet zijn voor het project. Liefst fulltime, maar werken met duidelijke afspraken en het naleven daarvan is ook mogelijk.

Testers hebben geen vrienden?
Een van mijn naaste collega's heeft als favoriete quote: "Testers hebben geen vrienden". Dat zegt hij met in het achterhoofd dat wij altijd onafhankelijk moeten zijn. Echter, ik kan mij nooit inhouden om te riposteren met "Da's vreemd, want de succesvolste projecten die ik heb gedaan kenmerkten zich altijd door een goede samenwerking tussen de teams". Agile testen heeft als uitdaging om beide zaken, onafhankelijkheid en goed samenwerken, effectief met elkaar verenigd te hebben.

De eindgebruikers spelen een belangrijke rol in de samenwerking: ze worden actief betrokken in het project om de acceptatietests te schrijven. Zij kunnen het beste bepalen op welke manier het systeem de meeste waarde voor de business oplevert. Het gaat dan uiteraard met name om de inhoud van de testcases, het automatiseren ervan is een teamverantwoordelijkheid. Naast een actieve rol tijdens de iteratie kunnen eindgebruikers ook een rol in de acceptatie per iteratie spelen. Door bijvoorbeeld op de opgeleverde, werkende software business scenario's uit te voeren.

Samenwerken in stappen:

- Samenwerking genereert feedback tussen de teamleden: het werkt kwaliteitsverhogend en creëert voortschrijdend inzicht.
- Samenwerking vereist eenvoud om de dingen voor iedereen begrijpelijk te houden.
- Samenwerking vereist ook flexibiliteit om de feedback te kunnen verwerken.

2.1.2. FEEDBACK

Feedback is letterlijk: terugkoppeling. Feedback werkt twee kanten op: het kan gegeven en ontvangen worden. Een gemiddeld softwareproject kent een fors aantal stakeholders: de opdrachtgever, productmanagement, functioneel beheer, technisch beheer, serviceafdeling, ketenpartners en sales, om er maar een paar te noemen. Feedback is essentieel. Het verzekert dat het systeem voldoet aan alle door de partijen gestelde eisen. In een agile project wordt de feedback vooral zíchtbaar gemaakt – werkende software als primaire voortgangsmeting. Daarmee wordt de status van het project over het algemeen beter bespreekbaar met de stakeholders, wat de uiteindelijke kwaliteit van het systeem verhoogt. In een traditioneel project wordt van stakeholders vooral verwacht dat ze vooraf alle eisen kunnen definiëren, in een agile project vervullen ze een actieve, meedenkende rol in het project. Helaas leert de praktijk ook dat lang niet alle stakeholders mee kunnen in het ritme en de frequentie van agile projecten. Ook een agile project vereist inzet en commitment van alle disciplines. Het is zaak om het doel en de weg er naar toe helder te communiceren. Om betrokkenheid en invloed van de stakeholders te verwachten. Een agile proces beoogt uiteindelijk wat we allemaal willen: kwaliteit op tijd.

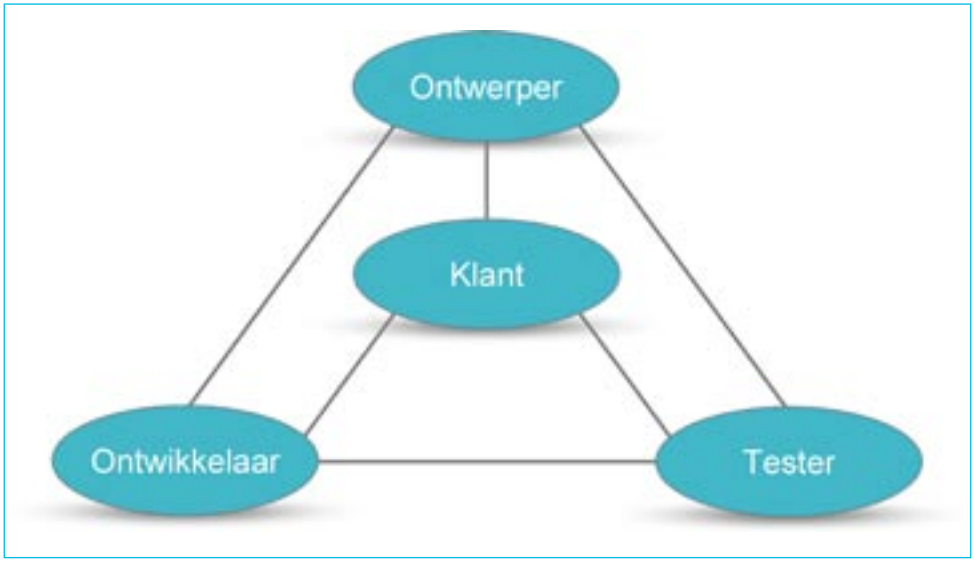
Binnen het team is feedback een belangrijke pijler. De lijnen van communicatie in een agile project zien er als volgt uit:

- De feedback tussen klant en ontwerper gaat over requirements. Welke zijn noodzakelijk en gewenst? Hoe kan het systeem de meeste waarde voor de business creëren?
- De feedback tussen klant en ontwikkelaar gaat over de (on)mogelijkheden van een gewenste technische uitvoering van een requirement.
- De feedback tussen klant en tester gaat over de voortgang en kwaliteit door middel van de uitgevoerde tests en de input van businesscriteria en -prioriteiten op de testscripts.

Problemen hebben de neiging om groter te worden naarmate ze langer onopgemerkt blijven.

- De feedback tussen ontwerper en ontwikkelaar speelt vooral op detail-technisch en functioneel niveau: welke technische oplossing voor de requirement is het meest eenvoudig en daarmee realiseerbaar en fit for purpose? Zijn er alternatieven voorhanden die wellicht beter aan de doelstellingen voldoen?
- De feedback tussen ontwerper en tester gaat om volledigheid en juistheid van de requirement, en het vaststellen van de grenzen aan de functionaliteit. Dit is de dialoog die testgoeroe Boris Beizer al wenste toen hij vaststelde "The design isn't done until the test cases are ready".
- Feedback tussen ontwikkelaars en testers speelt zich af op het gebied van unit- en systeemtests en de kwaliteit van de gerealiseerde software.

Zo krijg je dus een hele cyclus aan communicatie, die de kwaliteit van het systeem bevordert. Feedback is én zenden én ontvangen.



Afbeelding 2-2: Vier elementaire rollen in een agile softwareontwikkelproces

Voor de beeldvorming: hoe ziet een proces zónder feedback eruit?

- De klant kan het project niet bijsturen wanneer hij dat noodzakelijk acht.
- De ontwerper weet niet of de gewenste oplossingen ook technisch realistisch zijn.
- De ontwikkelaar weet niet of de software die hij ontwikkelt van voldoende kwaliteit is.
- De tester weet niet of zijn testscripts de technische valkuilen en businessrisico's afdekken.

Een van mijn statements is: Problemen hebben de neiging om groter te worden naarmate ze langer onopgemerkt blijven. In een proces met weinig feedback worden fouten vaak laat gevonden. Fouten herstellen die in een laat stadium gevonden zijn, is verhoudingsgewijs duur. Feedback verhoogt de kwaliteit en de voortgang van een proces. In een proces met veel feedback kun je zo laat mogelijk beslissen – pas dan immers heb je de meeste informatie. Daarom is het zo laat mogelijk beslissen óók een waarde van een agile team.

Zo laat mogelijk beslissen

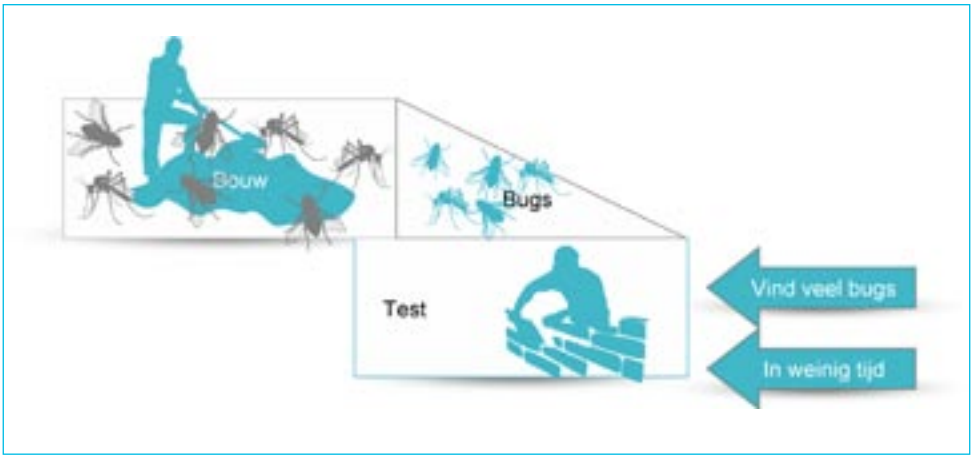
Eén van de gehanteerde credo's bij Autofabrikant Toyota is 'Decide as late as possible'[7]. Dit voorkomt zoveel mogelijk dat genomen beslissingen in een later stadium niet juist blijken te zijn en er veel tijd gestoken moet worden in het terugdraaien van die beslissingen. Ook designbeslissingen worden op die manier genomen: er wordt gewerkt met meerdere versies van één onderdeel. Op die manier kan er bij een eventuele afwijking op het laatst moment besloten worden welke van de versies het meest geschikt is. Traditioneel gezien zou er dan een heleboel beslissingen teruggedraaid en heroverwogen moeten worden, met alle bijkomende kosten van dien. Bij Toyota selecteren ze dan een van de andere beschikbare versies, zonder extra kosten.

Feedback verhoogt de kwaliteit en de voortgang van een proces.

Er is nog een factor van belang bij feedback. In een traditioneel project beginnen we pas met het uitvoeren van testen wanneer het voorgaande ontwikkelstadium volledig is uitgevoerd en voldoet aan de entry-criteria. Dan begint de periode waarin getest wordt, met als doel het vinden van fouten binnen de gestelde periode. Als het goed is wordt het aantal bugs naarmate de testtijd vordert steeds minder en wordt een systeem opgeleverd dat van voldoende kwaliteit is.

onmiskenbaar risico in: de ene fout wordt veroorzaakt door de andere, zogenoemde 'gevolgfouten'. Aan het eind van de bouwperiode krijgen we dus code die onnodig veel fouten bevat, omdat we niet eerder beginnen met het uitvoeren van testen.

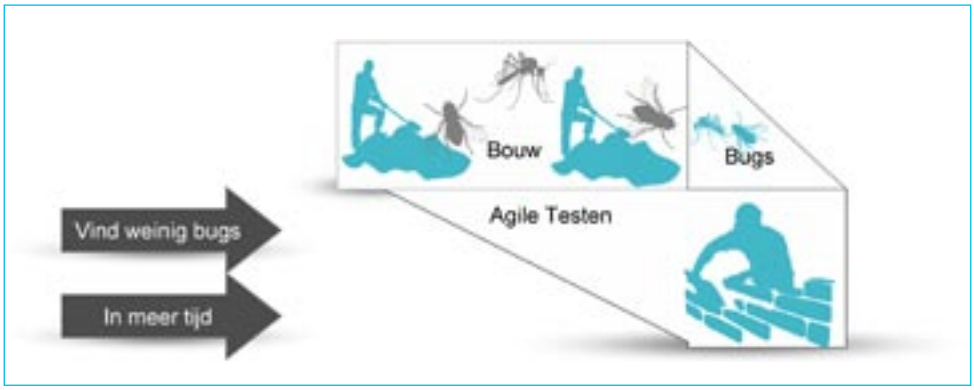
Een bekend gegeven in de ICT is de curve van Barry Boehm[8]: hoe vroeger het stadium is waarin een fout ontdekt wordt, des te goedkoper is het herstellen van de fout. Fouten die laat worden gevonden zijn dus relatief duur.



Afbeelding 2-3: Het herstellen van fouten uit de systeemtest in een traditioneel ontwikkelproces

Echter, doordat er veelal niet tijdens de bouw getest wordt maar erna, treedt er een ongewenst patroon op. Hoe goed je namelijk ook ontwikkelt, bugs zullen er altijd zijn. En doordat testen pas achteraf gebeurt wordt er na verloop van tijd software ontwikkeld op basis van code die nog niet getest is. Dit houdt een

Agile testen verschuift de testuitvoering bewust naar voren om de feedback te verhogen, en daarmee worden fouten in de code in een vroeger stadium gevonden. Op die manier worden kosten bespaard en – zoals hierboven beschreven – treden er ook nog eens minder fouten op.



Afbeelding 2-4: Bugherstel in een agile ontwikkelproces, over alle testsoorten heen

Agile testen wil fouten zo veel mogelijk voorkomen, door al tijdens de ontwikkeling te gaan testen. Op die manier wordt de negatieve spiraal doorbroken:

- Ontwikkelaars krijgen direct feedback op de code die zij ontwikkeld hebben, dus bugs worden in een zo vroeg mogelijk stadium geconstateerd.
- Daardoor wordt er geen code ontwikkeld op code die niet getest is, waardoor gevolgfouten voorkomen worden.
- Er worden dus minder fouten gemaakt, waardoor het testen efficiënter verloopt.
- Er is meer tijd beschikbaar om te testen - het ligt daarmee minder op het kritische pad.

De redenering van veel ICT-ers zal zijn: 'Ja maar, als je in een vroeg stadium gaat testen dan gaat dat dus ten koste van de voortgang van het ontwikkelen. Je moet immers fouten herstellen terwijl je nieuwe code moet ontwikkelen!' Maar dan kies je opnieuw voor productie in plaats van kwaliteit, en die rekening betaal je bij het testen, implementeren en in onderhoud nemen van het systeem.

"Simple, clear purpose and principles give rise to complex, intelligent behaviour. Complex rules and regulations give rise to simple, stupid behaviour." – Dee Hock, Ceo Visa

Feedback in stappen:

- Feedback vereist eenvoud omdat feedback kan leiden tot veranderingen. Je moet dus in staat zijn om wijzigingen snel door te voeren.
- Feedback vereist ook samenwerken, omdat de timing goed moet zijn.
- Feedback noopt tot flexibiliteit, omdat deze anders geen waarde heeft.

2.1.3. EENVOUD

Misschien is dit wel de moeilijkste waarde om goed te bevatten: we zijn immers hoog opgeleide mensen die geleerd hebben om goed na te denken voor we ergens aan beginnen. Om onze aannames goed te documenteren, zodat er zorgvuldige verslaglegging plaatsvindt. Wat we ten ene male proberen te voorkomen is dat we ergens fouten maken gedurende de reis. Specifiek voor testen geldt dat we graag volledig willen zijn – in detail uitgewerkte plannen en scripts moeten hiervoor zorgen. Vaak bestaan templates van mastertestplannen – nog niet ingevuld – uit een pagina of veertig. Het lijkt soms wel of we het credo 'meer is meer' aanhangen!

Agilisten gaan hier anders mee om. Want wat zijn de vervelende eigenschappen van het volledig willen zijn en het willen voorkomen van fouten? Dit proces kost vaak veel tijd kost en is bijzonder complex. Als daar fouten in gemaakt worden, is het vaak tijdrovend om ze op te lossen. Zeker als het met terugwerkende kracht moet worden gedaan, en vaak kan het dan alleen nog maar door de specialist gebeuren. Agile gaat er vanuit dat wanneer je ergens mee begint, je nog niet exact weet wat het eindproduct zal zijn. En dus dat er wijzigingen gedurende de weg zullen optreden. Daarmee wordt de onderhoudbaarheid van hetgeen waar je mee bezig bent een belangrijke kwaliteitseigenschap. Daarnaast is het kunnen samenwerken een belangrijk uitgangspunt voor het streven naar eenvoud. Alles zal door iedereen in het team snel begrepen moeten kunnen worden. Waar je ook mee bezig bent, je zult dus eenvoud na moeten streven. Om het wijzigbaar en samenwerkingsgericht te maken.

Voor de praktijk betekent dit dat je een bottom-up aanpak nastreeft: je vertelt wat je nu weet, op basis van de inzichten van vandaag. Niet te veel willen voorspellen maar werken vanuit het heden. Voor veel testers zal dit wennen zijn: voor ons is een document pas af en goed als het volledig is. Maar voor een agilist is een document nooit af, en moet het altijd leesbaar en direct uitvoerbaar zijn.

Eenvoud in testscripts
In een project vroeg ik aan een senior-tester of hij een sjabloon voor het testontwerp van de systeemtest wilde maken. Hij had enkele jaren ervaring in het testen en hij wist dat het project de nodige functionele onduidelijkheden kende, en veranderingen geaccepteerd moesten worden. Na een dag of twee kreeg ik de eerste uitwerking: werkbladen met kwaliteitscriteria, prioritering, relatieve belangen, systeemmodulen, onderverdeling daarvan in objecten en daar dan weer gewichten aan gehangen. Hartstikke mooi, heel veel denkwerk alleen… amper bruikbaar in een context waarin veel wijzigingen zullen optreden. Ik heb hem toen de vraag gesteld: "Hoeveel tijd heb jij nodig om in dit sjabloon tien testgevallen te wijzigen?" Hij knikte instemmend – en we hebben besloten ons tot de kern van de testgevallen te beperken.

Eenvoud in stappen:

- Eenvoud is het gevolg én de oorzaak van feedback: de werkwijzen dienen voor iedereen begrijpelijk te zijn, en de onderhoudbaarheid van de producten dient hoog in het vaandel te staan.
- Eenvoud versterkt samenwerken doordat er minder miscommunicatie optreedt.
- Eenvoud maakt flexibiliteit mogelijk doordat eenvoudige dingen makkelijker wijzigbaar zijn.

2.1.4. FLEXIBILITEIT

In de Engelse taal kan een dier agile zijn, bijvoorbeeld een poema. Of een auto: "this Lotus is quite nimble and agile". Dit wil zeggen dat het dier of de auto zich makkelijk aanpast aan moeilijke omstandigheden en op veel soorten terrein uit de voeten kan. Flexibiliteit als randvoorwaarde dus om doelgerichter te kunnen zijn.

In softwareontwikkeling en testen ligt het iets anders, maar toch zijn er veel overeenkomsten. De belangrijkste reden om flexibiliteit na te streven is: het opleveren van meer waarde voor de business. Bijvoorbeeld door eerder een werkend product op te leveren en zo te beginnen met het terugverdienen van de investering. Maar ook waarde voor de gebruikersorganisatie door hun actieve participatie

"Learn from yesterday, live for today, hope for tomorrow. The important thing is to not stop questioning." – Albert Einstein

te gebruiken voor het ontwikkelen van functionaliteit die ze echt nodig hebben en waar ze goed mee kunnen werken. Wanneer je nauw samenwerkt en veel feedback in je proces inbouwt, is flexibiliteit ook een must – wat ga je anders met de feedback doen?

Het bereiken van flexibiliteit in het testen kan op diverse manieren:

- Door eenvoud na te streven: veel details betekenen vaak veel rework als er iets wijzigt.
- Door het sjabloon van het testontwerp zodanig te maken dat de tester het flexibel kan inzetten (niet gedwongen is slechts één manier van werken te hanteren).
- Door niet te veel vooruit te willen plannen: beperk je scope bij voorkeur tot één iteratie.
- Door documenten zo veel mogelijk generiek te houden en details pas binnen de iteratie af te stemmen (en zonodig te documenteren).
- Door de rol van de tester flexibel in te vullen: niet altijd is de tester de meest geschikte persoon om iets te testen – dat kan soms ook een klant, gebruiker, ontwerper of ontwikkelaar zijn.

Flexibiliteit in stappen:

- Flexibiliteit in testen is mogelijk wanneer je uitgaat van samenwerken, eenvoud en feedback.
- Flexibiliteit is een uitstekend uitgangspunt om waarde te creëren voor business en gebruikers.
- Flexibiliteit gaat uit van een proces waarin veel feedback plaatsvindt en er nauw samengewerkt wordt.

2.2. OVEREENKOMSTEN EN VERSCHILLEN MET TRADITIONEEL TESTEN

De hierboven geschetste agile normen en waarden zorgen logischerwijs voor een andere cultuur. In een agile proces is het team eindverantwoordelijk voor de producten die ze oplevert. Tekortkomingen of kwaliteitsproblemen die aan het eind van de iteratie aan het licht komen, zijn daarin geen fouten van individuen maar van het hele team. Blijkbaar is het proces niet goed uitgevoerd waardoor er fouten zijn ontstaan. Er wordt niet gekeken naar wiens verantwoordelijkheid het is. De vraag is meer: hoe voorkomen we dat we de volgende keer weer dezelfde fout maken? Doordat iteraties kort duren worden fouten snel opgespoord én dient zich direct de mogelijkheid aan om het te verbeteren. De leercurve van een agile team is daarmee extreem kort te noemen: verbeterslagen kunnen per maand gemaakt worden.

In een agile team is een tester dus geen solist: testen en de kwaliteit van het testen, is een team issue. Als de situatie dreigt dat een tester het noodzakelijke testen van een iteratie niet meer tijdig kan afronden, dan zullen er keuzes gemaakt moeten worden:

- Er wordt vanuit het team bijgesprongen om de tester te ondersteunen.
- Er wordt minder functionaliteit ontwikkeld om de tester de gelegenheid te geven alle benodigde inspanning te leveren.
- Er wordt overlegd met de product owner of de requirements met minder diepgang getest kunnen worden.

Voor testers biedt agile ontwikkelen zeer veel voordelen:

- Door de nauwe samenwerking ontstaat er al snel een compleet beeld van het systeem en het project.
- De verantwoordelijkheid van testen en dus de kwaliteit van het systeem ligt bij het team, daarmee wordt voorkomen dat testen er laat bij betrokken wordt of iets is wat er nog aan toe moet worden gevoegd.
- Het testen is geïntegreerd in het ontwikkelproces en genereert daarmee belangrijke feedback aan het ontwikkelteam. Testers worden in een vroeg stadium bij beslissingen betrokken.
- Samenwerken met de klant betekent ook dat de acceptatietest continu met de klant wordt afgestemd. En er dus – als het goed is – geen onaangename verrassingen meer optreden bij in productiename.

Toch zijn er diverse overeenkomsten tussen traditioneel testen en agile testen:

- Mastertestplanning: het is nog steeds nodig om een generiek document te gebruiken dat de teststrategie en haar consequenties beschrijft. Dit wordt nader uitgewerkt in paragraaf 2.4.
- Testtechnieken: ook testtechnieken moeten worden gebruikt, net zoals traditioneel testen doet op basis van een risicomatrix. Meer hierover in paragraaf 2.6.

- Gebruik van testscripts: de geselecteerde testtechnieken worden uitgewerkt naar testgevallen in de testscripts. Een agile team bereidt zich wel degelijk voor voordat het aan de slag gaat. Ook een agile tester stelt testscripts op en ziet het realiseren van (test-)documentatie als een integraal onderdeel van de iteratie.

Daarnaast kunnen de volgende verschillen genoteerd worden:

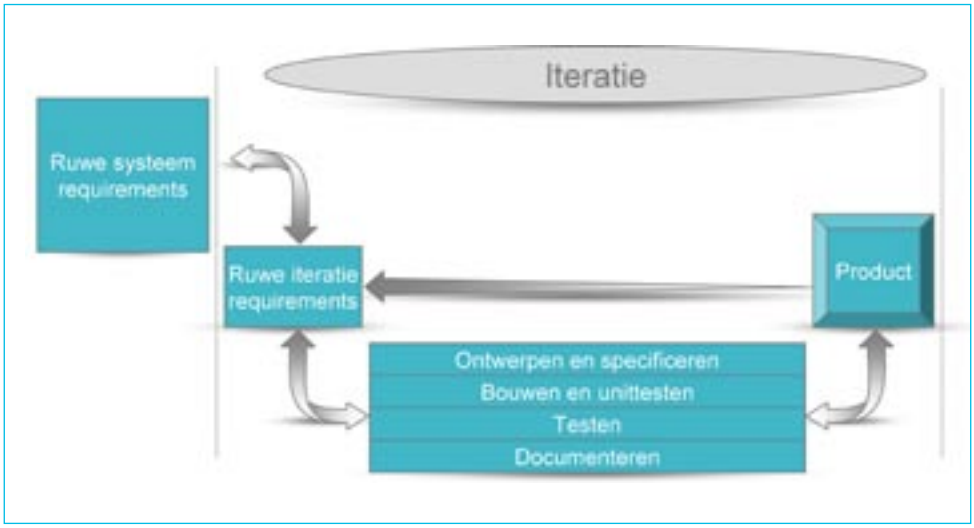
- Agile testen is multidisciplinair teamwork. Binnen het traditioneel testen is het teamwork vooral binnen het testteam zelf van toepassing.
- Agile testen gaat uit van het in een zo vroeg mogelijk stadium uitvoeren van testen. Traditioneel testen begint pas met het uitvoeren van testen als het systeem uitontwikkeld is.
- Binnen agile testen is er geen sprake van een strikte fasering van testactiviteiten. Bij traditioneel testen is dat vaak wel het geval.
- Agile testen is gebaseerd op ruwe requirements en gaat uit van samenwerking binnen het team en met de klant om tot de juiste, volledige testgevallen te komen. Traditioneel testen gaat uit van volledige, juiste, consistente documentatie.
- Agile testen gaat uit van het halen van de deadline (de iteratie) en stelt zonodig in overleg de procedures bij. Traditioneel testen gaat uit van de afgesproken procedures en werkwijzen en zal bij onvolkomenheden uitloop melden.
- Gezien het iteratieve karakter van het project zal er binnen agile testen vaak en veelvuldig gebruik worden gemaakt van tools en geautomatiseerd testen. Binnen traditioneel testen ligt die afweging anders.

2.3. ITERATIEF WERKEN

Voordat we nu de diepte ingaan met het agile testen, eerst een introductie in iteratief werken om een helder beeld van de context te krijgen. Veel van de principes van agile testen hebben hun grondslag in het iteratieve ontwikkelproces.

2.3.1. ITERATIE

Een iteratie betekent letterlijk: herhaling. Voor softwareontwikkeling houdt dit in dat voor iedere requirement vrijwel het gehele ontwikkelproces doorlopen wordt. Feitelijk is er in een watervalproject sprake van één iteratie, omdat er in principe geen herhaling van zetten wordt uitgevoerd. Vaak wordt er dan uitsluitend aan het eind van het project geëvalueerd waardoor je uitsluitend op projectbasis een lerend effect kunt krijgen. In het volgende project kun je het dan beter of anders doen. In agile ontwikkelen wordt dit



Afbeelding 2-5: Bij de start van de iteratie wordt een set aan requirements vastgesteld. Vervolgens worden deze in een integraal proces ontworpen, ontwikkeld, getest en gedocumenteerd. Aan het eind van de iteratie wordt een werkend product opgeleverd. Er zijn hierin diverse feedbackloops.

lerende effect als uitgangspunt genomen: door vaak dezelfde procesmatige stappen te zetten kun je snel leren om iets goed te doen. Iedere iteratie is daarmee een integrale stap in het ontwikkelen van de software. De meest kritische stappen: detailontwerp, bouw, test en documentatie, worden als één geheel beschouwd. Het proces staat daarbij centraal, en het te ontwikkelen systeem is min of meer het lijdend voorwerp. De requirements passeren de revue en zijn niet meer dan een onderdeel van het geheel.

2.3.2. TIMEBOX

Een iteratie is per definitie één timebox. Dat wil zeggen dat de deadline vaststaat en er geen mogelijkheid tot verschuiven is. Door middel van het prioriteren van functionaliteit wordt het proces flexibel gemaakt. Dit biedt het project de mogelijkheid om andere belangrijke variabelen: tijd, budget en kwaliteit, wél te fixeren. Zodat het einde van de timebox eindigt in geteste, goede software die binnen budget is gebleven. Als de product owner de prioriteiten juist heeft gesteld, wordt er software opgeleverd die van hoge waarde is voor de business.

In het fixeren van de timebox moet een agile team rigide zijn, om de volgende redenen:

- Er is absolute duidelijkheid voor het hele team wanneer iets af moet zijn.
- Er wordt altijd op tijd opgeleverd.
- De software kan altijd van de afgesproken kwaliteit zijn omdat dit niet flexibel gemaakt wordt.
- Hiermee kan vanaf het tijdstip van oplevering waarde voor de business worden gerealiseerd.

Natuurlijk heeft dit wel wat consequenties: zo moet de flexibiliteit die je wenst wel ergens vandaan komen. In een agile project ligt die flexibiliteit in het aantal requirements dat per iteratie wordt opgepakt. Een requirement wordt ook wel user story genoemd.

De requirements die op de rol staan voor de iteratie dienen allemaal geprioriteerd te zijn. Vanuit de agile methode DSDM kan hiervoor het MoSCoW principe worden gebruikt. Dit acroniem staat voor het prioriteren volgens:

- Must Haves, voor het systeem fundamentele requirements. Ze bepalen het minimaal aantal noodzakelijke items die aan het eind van de iteratie moeten worden opgeleverd om een werkend systeem met voldoende waarde voor de business te leveren.
- Should Haves, requirements die belangrijk zijn om op te leveren in de iteratie maar die niet essentieel voor de geleverde business waarde zijn.
- Could Haves, requirements die op de korte termijn gemist kunnen worden.
- Want to Haves, requirements die op de wachtlijst staan.

Nu gebeurt het nogal eens dat in eerste instantie de business alle requirements als Must Haves bestempelt (of als 'prio 1'). Men streeft dan naar een functioneel 'perfect' systeem. Maar dan staat het agile team opnieuw voor de keuze welke van de andere belangrijke parameters tijd, kwaliteit of budget flexibel te maken. Dit om eventuele tegenvallers op te vangen. Wanneer deze situatie zich voordoet, bedenk dan het volgende: 'Alles is belangrijk, maar niet alles is even belangrijk'.[9]

In deze structuur is de business dus verantwoordelijk voor de prioriteit van de requirements. Maar in het definitief toekennen ervan moet er uiteraard eerst afgestemd worden met het ontwikkelteam: het moet technisch mogelijk en relevant zijn om een bepaalde set aan requirements in een iteratie te ontwikkelen. De business komt dus met een voorstel, er vindt overleg met het team plaats en dat resulteert in een definitieve set requirements voor de iteratie.

Wat mag de business nu verwachten van de productiviteit van het team? De ervaring leert dat er in het begin van uitgegaan kan worden dat ongeveer zestig procent van de requirements voor de iteratie een Must Have is. Wanneer het team dat haalt, dan scoren ze in rapporttermen een zes. Circa twintig procent van de overige requirements kunnen Should Haves zijn, en de resterende twintig procent zijn dan Could Haves. Als buffer voor leegloop bij een onverwacht hoge productiviteit kunnen de Want to Haves aangesproken worden. De ervaring leert overigens dat uitzonderingen aan beide zijden voorkomen: een team haalt aan het eind van de iteratie niet de beoogde Must Haves, maar met hetzelfde gemak zijn er ook situaties voorgevallen waarbij het team aan de business nog wat extra requirements vroeg omdat alles al af was. Dat zijn leermomenten, en agile teams leren snel. Geef sowieso ieder nieuw agile team minimaal drie iteraties de tijd om het proces op elkaar af te stemmen. Om het leereffect te versnellen zou er een pilot gehouden kunnen worden met een kleiner team en minder requirements, waarin de lengte van de iteratie gehalveerd wordt.

2.3.3. OPLEVERING

Aan het eind van de iteratie, en dus de timebox, dient werkende software opgeleverd te worden. Het liefst software die de hele cyclus heeft doorlopen, dus inclusief een deployment. Het mooiste is installeerbare software op te leveren, dus inclusief eventuele conversieprogrammatuur, gebruikersdocumentatie, systeemdocumentatie en installatiehandleiding. Dat klinkt wellicht vreemd wanneer het systeem nog niet uitontwikkeld is, maar het is wel dé manier om meteen waarde voor de business op te leveren. De klant kan er dan mee aan de slag. Dit is vanuit testperspectief een fikse uitdaging: in iedere iteratie wil je alle testsoorten uitgevoerd hebben. De belangrijkste technische risico's moeten immers frequent getest worden. Voor de lagere testsoorten

"Perfectie wordt niet bereikt doordat je niets meer kunt toevoegen, maar doordat je niets meer kunt weglaten" – Antoine de Saint-Exupéry

(zoals unit- of integratietest) is dit niet zo'n probleem, dat kan vaak geautomatiseerd en in de eigen ontwikkel- of testomgeving. Maar voor grotere projecten in grotere bedrijven met meerdere omgevingen is het uitvoeren van een acceptatietest op de productiegelijke omgeving een kluif. Vaak wordt deze samen met de productieomgeving extern beheerd en zijn de contracten niet toegespitst op het leveren van ondersteuning aan agile projecten. Het actief bij het project betrekken van alle partijen is dan een randvoorwaarde.

Daarnaast, en dat is in sommige softwarebedrijven zeer belangrijk, wordt er op die manier een basis van vertrouwen gelegd tussen sales en ontwikkeling: het team toont aan dat het in staat is om frequent saleswensen om te zetten in een werkend systeem. Wanneer dit proces goed werkt, wordt de druk op het ontwikkelteam verlegd naar het salesteam – zij zullen moeten aantonen dat de geleverde software ook daadwerkelijk de meest waarde oplevert. Hoewel de aanzet lastig en soms listig is, genereert het geven van feedback door werkende software een basis van vertrouwen.

Aan het eind van de iteratie moeten minimaal de volgende producten opgeleverd worden:

- Werkende, geteste software
- Testresultaten
- Systeemdocumentatie
- Gebruikersdocumentatie

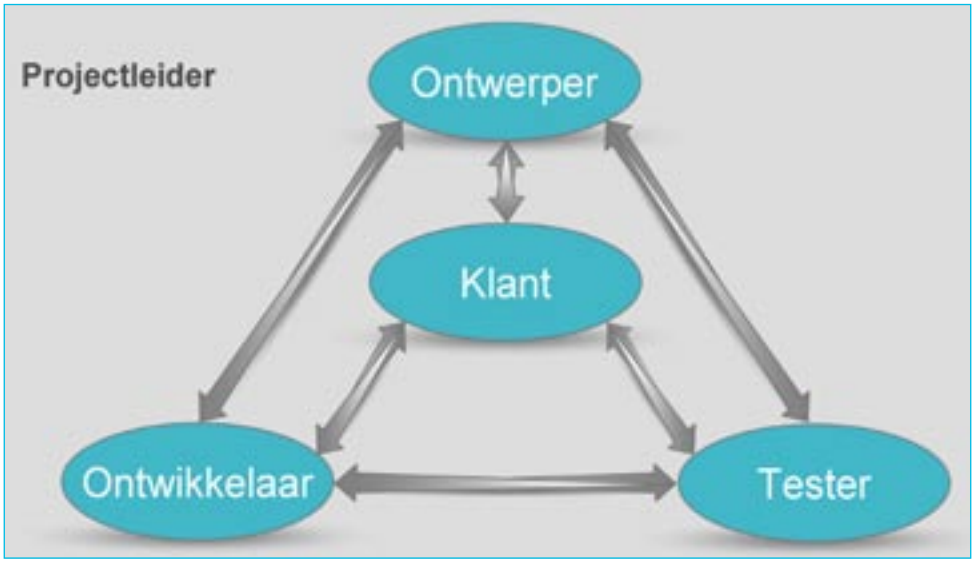
Dit laatste houdt in dat ook documentatiespecialisten opgenomen moeten worden in het agile team en mee moeten draaien in de iteratieve werkwijze.

2.3.4. PROJECTMANAGEMENT

Iteratief werken heeft nogal wat consequenties voor projectmanagement. Het gaat te ver om in dit boek alle details te bespreken, daarom de belangrijkste highlights.

In een cultuur van samenwerken, flexibiliteit en partnership met de klant is de traditionele rol van de projectmanager die bepaalt en controleert, niet langer meer passend. Veel van zijn traditionele verantwoordelijkheden zijn gedelegeerd naar het team. De projectleider zal veel meer de rol van een facilitator op zich nemen: iemand die het team in staat stelt om tot grote hoogten te stijgen. Iemand die blokkades uit de weg ruimt en in staat is om als gids te dienen. Kortom: van een agile projectleider wordt vooral dienend leiderschap verwacht. Omdat dit andere competenties van een projectleider impliceert, is die rol niet voor iedere traditionele projectleider weggelegd.

Van nature zal een projectleider gewend zijn om met verschillende rollen om te gaan. Maar de agile projectmanagementmethode Scrum[10] gaat er uit van twee: de rol van team en de rol van product owner. Het team is het voltallige



Afbeelding 2-6: De faciliterende rol van agile projectleider; hij heeft geen directieve taak, maar een dienende.

team dat in de iteratie meedraait. De product owner is feitelijk de klant: degene die de input levert voor de requirements die de iteratie ingaan. In de methode Scrum wordt de iteratie ook wel sprint genoemd. De projectleider zelf heet in Scrum-termen de ScrumMaster[11]. De product owner bepaalt de prioriteit van een requirement, waarbij het wel belangrijk is dat ook het team hierop input kan leveren omdat er ook een technische factor in zit. Een serie requirements moet immers ook technisch hebben een zekere samenhang. Het is de taak van de projectleider om de communicatie tussen de product owner en het team te faciliteren. Daar komt vaak ook nog bij dat de klant de spelregels van agile software ontwikkeling nog moet leren. Hierin schuilt een coachende taak. Zo zal de projectleider ook met de product owner om tafel zitten om over een aantal iteraties heen te kijken wat er basaal aan nieuwe functionaliteit gewenst is.

Omdat er in principe per iteratie nieuwe requirements gepland kunnen worden, doet een agile team bewust niet zo heel veel aan planning. Door de iteratieve werkwijze liggen immers de releasedata al vast. En de klant bepaalt de prioriteiten. Maar er moet natuurlijk wel iets verteld kunnen worden: wat is de productiviteit van het team? Hiervoor wordt er per iteratie gemeten wat er opgeleverd wordt. Sowieso wordt het verschil gemeten tussen de geplande tijd per requirement en de daadwerkelijk bestede tijd per taak. Dit vindt niet plaats in een context van een afrekencultuur maar in een context van een leercultuur. Als de planning niet blijkt te kloppen is het geen fout in de uitvoering maar in de schatting. Daarnaast wordt het aantal opgeleverde requirements per prioriteit gemeten.

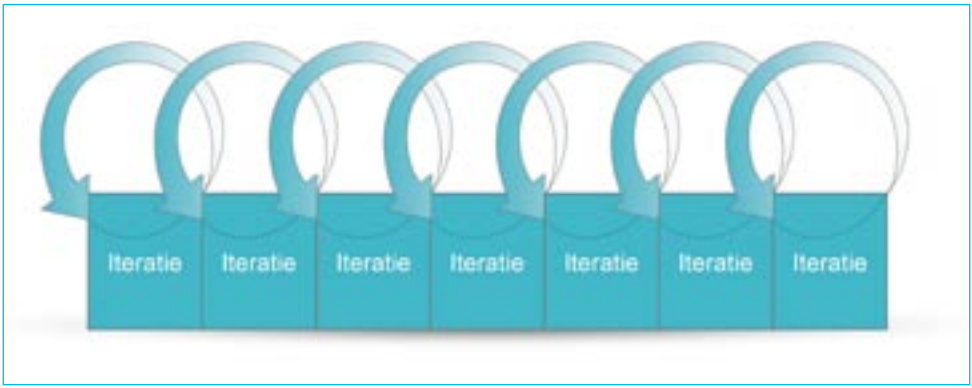
We spreken in agile termen dan vaak over de velocity: de snelheid waarmee een agile team een requirement integraal kan opleveren. Dat is dus het detailontwerp, de ontwikkeling, het testen en het documenteren.

2.3.5. REGRESSIETEST

Door de iteratieve werkwijze zal er per iteratie een regressietest* moeten worden uitgevoerd. In principe wordt dit opgevangen door in de iteratie ook meteen geautomatiseerde testscripts te ontwikkelen. Dit kan door de tester of door de ontwikkelaars gedaan worden, of door een gespecialiseerde testengineer die dan uiteraard ook in de iteratie meedraait. Niet alle testgevallen hoeven geautomatiseerd te worden, dit kan afhangen van de risicocategorie van de functionaliteit. Door de hoge snelheid waarmee er ontwikkeld wordt en er dus wijzigingen in de software optreden, is het automatiseren van de regressietest van groot belang. Een geautomatiseerde testset is hiervoor namelijk een uitstekend vangnet. Het goed opzetten van geautomatiseerd testen is een vak apart[12].

Wanneer er geen mogelijkheid tot automatiseren is of wanneer de niet-geautomatiseerde functionaliteit hertest moet worden, kan voor de oplossing van het prioriteren van de testcases gekozen worden. Dan worden de belangrijkste testgevallen het eerst uitgevoerd. Omdat er veelal toch in enige mate handmatig getest wordt, kan er op die manier voorkomen worden dat het overzicht op de prioriteiten verloren wordt.

* Een regressietest toont aan of de nieuwe functionaliteit al dan niet impact heeft gehad op de ongewijzigde code.



Afbeelding 2-7: De leercurve van een agile project: iedere iteratie opnieuw.

2.3.6. ITERATIE IN EEN SCHEMA

Per iteratie is er sprake van een 'nieuwe ronde'. Het volledige proces van detailontwerp – bouw – test – documentatie vindt één keer plaats voor geprioriteerde requirements. De klant kan er voor kiezen om een requirement waar hij toch niet tevreden over is, nogmaals in te brengen in de volgende iteratie. In theorie is het zelfs zo dat álle requirements opnieuw de iteratie in kunnen gaan, bijvoorbeeld om hetzelfde systeem in een nieuwe ontwikkelomgeving te ontwikkelen. Maar het is nog maar de vraag of er dan nog sprake is van voldoende waarde voor de business.

Aan het eind van de iteratie vindt altijd een evaluatie of retrospective plaats: aan de teamleden en de klant wordt gevraagd wat de leermomenten, verbeterpunten en wat de 'dit moeten we blijven doen' punten zijn. Op die manier kan er namelijk heel snel geleerd worden: per iteratie kunnen er verbeteracties plaatsvinden.

Aan het eind van de iteratie vindt er ook een oplevermoment plaats, waarbij minimaal wordt opgeleverd:

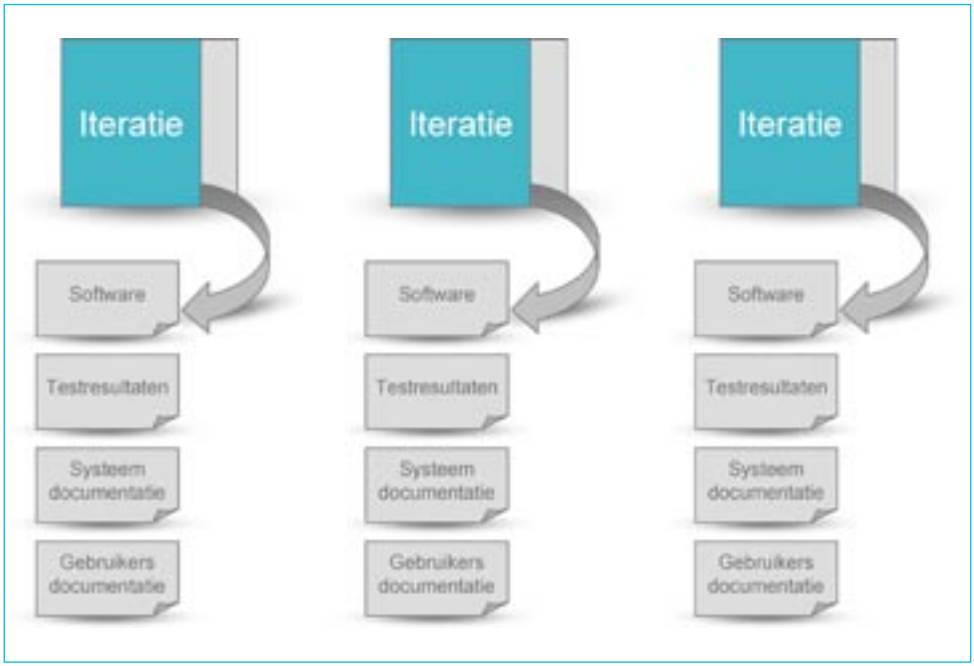
- Werkende software (daarbij is het testen inbegrepen)
- Testresultaten
- Systeemdocumentatie
- Gebruikersdocumentatie

Daarnaast zouden er ook nog additionele zaken van toepassing kunnen zijn, zoals testresultaten van de performancetest, integratietest of contextgevoelige helpfuncties. Na afronding van de iteratie vindt de start van de nieuwe iteratie plaats: hierin worden de nieuwe requirements beoordeeld en de risico-analyse gedaan. Meestal zit er een of twee dagen tussen de iteraties. Dit om de voorgaande iteratie correct af te ronden en een evaluatie te doen. Daarna worden de voorbereidende activiteiten voor de nieuwe iteratie gestart.

Net zoals in een traditioneel project zal de klant moeten bepalen wat voor hem 'af' is, dus formeel bepaalt hij de acceptatiecriteria. In principe zijn die criteria voor alle iteraties hetzelfde. Dat betekent dus ook dezelfde criteria voor een iteratie die niet in productie wordt genomen. Wanneer er namelijk verschillende acceptatiecriteria voor iteraties worden gehanteerd, wordt het principe van agile ontwikkelen aangetast. Niet alle noodzakelijke activiteiten worden dan per stap doorgevoerd. Dit is feitelijk uitstel van werk en dus uitstel van noodzakelijke feedback. Dit kan tot gevolg hebben dat er op het laatste moment nog vermijdbare fouten worden geconstateerd. Een alternatief scenario hiervoor is een speciale 'acceptatie-iteratie' introduceren. Daarin kunnen expliciet de acceptatie-eisen als taken worden ingepland als onderdeel van de scope.

2.4. EEN AGILE MASTERTESTPLAN

Traditioneel wordt het mastertestplan aan het einde van de informatieanalysefase geschreven, wanneer de meeste requirements verzameld zijn en er een start wordt gemaakt met het functioneel ontwerp. Omdat een agile project niet een dergelijk lange 'aanloop' kent zal een mastertestplan vanaf de eerste iteratie beschikbaar moeten zijn. Omdat er per iteratie een volledige ontwikkelcyclus wordt doorlopen, moeten dan de diverse testsoorten al bekend zijn. Daarbij moet er ook globaal bekend zijn hoe de testsoorten uitgevoerd gaan worden. Dat hoeft niet tot in alle details, een agile team maakt immers gebruik van voortschrijdend inzicht. Iets toevoegen aan een bestaande situatie is geen eerdere fout, maar een nieuwe, waardevolle toevoeging. Dus nog meer dan in een traditionele omgeving is een agile mastertestplan een levend document. In principe wordt het document geschreven door de tester en de projectleider samen, in nauwe afstemming met het voltallige team.



Afbeelding 2-8: Iedere iteratie resulteert in software, testresultaten, systeem- en gebruikersdocumentatie

Een agile team ontwikkelt uitsluitend documentatie die waarde toevoegt voor de klant. Iedere agile tester zal zich dus moeten afvragen of de traditionele verdieping naar detailtestplannen noodzakelijk is. Als dat document geen nieuwe mogelijke risico's afdekt of er geen sprake is van een contractuele verplichting dan zal die beslissing negatief uitvallen. Dit kan ook het geval zijn als het team klein is en er geen sprake hoeft te zijn van elkaar verkeerd begrijpen. Als het team groter is, of over meerdere locaties verspreid is dan zal die noodzaak groter zijn. Wat vanuit een agile perspectief meer waarde toevoegt is een detailtestplan per iteratie. Daarin wordt per iteratie beschreven welke functionaliteit getest

wordt en welke criteria daarvoor gelden. Omwille van de overzichtelijkheid en leesbaarheid dient dat dan wel in maximaal twee A4-tjes te gebeuren.

2.4.1. PLANNING

In principe draait de tester fulltime mee in het team. Maar uiteraard moet er een zinvolle verdeling naar rol plaatsvinden. In ieder geval gebeurt dit ook weer per iteratie: er wordt ingeschat hoeveel testtijd er besteed moet gaan worden en hoeveel mensen daarvoor nodig zijn. Kleine tekortkomingen of overlappingen kunnen worden opgevangen door eigen teamleden: een tester kan unittests* reviewen, geautomatiseerde scripts maken of documentatie schrijven. Zo kunnen ook andere teamleden testtaken overnemen wanneer er een tekort aan testcapaciteit is.

* Een unittest toont aan of de ontwikkelaar de logica van de applicatie technisch correct heeft geïmplementeerd. Deze wordt per component uitgewerkt. Unittests worden frequent in een geautomatiseerd framework gedraaid als een vorm van regressietest.

2.4.2. TESTSTRATEGIE

Onderdeel van het mastertestplan is het opstellen van een teststrategie. Traditioneel bestaat deze uit een drietal achtereenvolgende stappen:

1. Het uitvoeren van een productrisicoanalyse samen met de stakeholders.
2. Het opstellen van een risicomatrix.
3. Het door de testmanager of testcoördinator uitwerken van de risicomatrix naar diepgang, testsoorten en testtechnieken.

Vanuit een agile perspectief wordt een eenvoudiger model gehanteerd. Eenvoud eerst. Allereerst vindt er een stap risico-inschatting plaats die de eerste twee traditionele stappen samenvoegt. Vervolgens wordt op basis daarvan wel een vertaling naar testtechnieken gemaakt, maar dit gebeurt pas in de iteratie.

2.4.3. RISICO-INSCHATTING

De risico-inschatting is het indelen van de requirements naar risicocategorie. Omdat een agile team naar eenvoud en zoveel mogelijk feedback streeft, wordt dit met het hele team gedaan inclusief de belangrijkste persoon: de product owner. Hij is verantwoordelijk voor de eerste parameter van de risico-indeling: de business impact, oftewel de verwachte schade. Het team is verantwoordelijk voor het bepalen van het technisch risico, oftewel de foutkans.

De uitkomst wordt per requirement vastgelegd en er volgt een inschatting in de risicocategorieën I, II, III of IV. Over het algemeen is deze lichtgewicht aanpak voldoende voor een agile team. Als het team er niet uitkomt kan er gekeken worden naar de traditionele aanpak, waarin er eerst een productrisico-analyse wordt gedaan en vervolgens een vertaling naar een risicomatrix plaatsvindt.

Afbeelding 2-9: een risicomatrix: systeemdelen of requirements worden beoordeeld op hun risicofactor

2.4.4. BEPALING VAN DE TESTTECHNIEKEN

In een agile team kan er tijdens de iteratie het een en ander veranderen. Testers dienen dus rekening te houden met:

- de requirements, die zijn ingedeeld naar prioriteit. Het kan zijn dat niet alle items aan het eind ook daadwerkelijk worden opgeleverd. Uiteraard streeft het team er wel naar en mag er verwacht worden dat alle Must Haves (of 'prio's 1') opgeleverd worden.
- het detailontwerp, dat binnen de iteratie wordt afgehandeld. Er gaan relatief 'ruwe' requirements de iteratie in. Dit betekent dat technische oplossingen nog ontwikkeld moeten worden, op basis van voortschrijdend inzicht kunnen wijzigen of simpelweg kunnen vervallen.

Dit heeft als gevolg dat de agile tester voor een dilemma komt te staan:

- Richt ik mijn testinspanning vooral op de Must Haves – met het risico dat minder belangrijke requirements met wellicht een hoge risicocategorie niet (goed) getest worden?
- Richt ik mijn testinspanning vooral op de requirements met een hoge risicocategorie – met het risico dat ik tijd aan het besteden ben aan zaken die niet opgeleverd gaan worden?

Daarnaast is het zo dat agile teams het liefst zo laat mogelijk beslissen. Niet omdat beslissen niet kan, maar omdat het team er bewust voor kiest dat niet te doen. Immers, pas op het laatste moment is de meeste informatie beschikbaar. Maak gebruik van voortschrijdend inzicht. Het nemen van vroege besluiten, sluit het opdoen van leermomenten uit.

Daarom zal een agile tester de vertaling van de risicocategorie naar testtechnieken pas op het laatste moment doen: op het moment dat hij begint met het opstellen van het testontwerp. Dan heeft hij feedback ontvangen van de ontwerper en de ontwikkelaar. In principe moet dan ook het testontwerp zo 'open' mogelijk zijn, dus niet teveel dwingen richting een testtechniek. De keuze die de tester dan maakt is gebaseerd op de volgende variabelen:

- Welke prioriteit heeft deze requirement?
- Welke risicocategorie heeft deze requirement?
- Welke testtechnieken zijn geschikt om hierbinnen het gewenste kwaliteitsniveau te behalen?

Een hulpmiddel hierbij is het gebruik van Quick Reference Cards (QRC), dit wordt nader uitgewerkt in paragraaf 2.6.1.

2.5. INTAKE TESTBASIS

Het is een mythe dat agile teams geen documentatie gebruiken. Het is geen mythe dat agile teams het liefst zo min mogelijk documentatie gebruiken. Dat heeft de volgende motivaties:

- Traditioneel gezien worden in documenten besluiten genomen die later vaak moeilijk omkeerbaar zijn, bijvoorbeeld ten aanzien van architectuur, ontwerp en teststrategie. Een agile team wil juist flexibel zijn.
- Besluiten die in die documenten genomen zijn, blijken gedurende het project vaak ingehaald te worden door de actuele stand van zaken. Een agile team beslist het liefst zo laat mogelijk.
- Documenten zijn een abstractie van de werkelijkheid. De werkelijkheid is de werkende software. Een agile team werkt liever vanuit de werkelijkheid.

- In veel projecten worden documenten geschreven omdat dat vanuit het proces of de contracten gevraagd wordt, echter de meerwaarde ervan voor de klant wordt zelden in de afweging meegenomen. Een agile team wil zoveel mogelijk waarde voor de business leveren, en zal zich dus ook mét de klant afvragen of het opleveren van een document toegevoegde waarde heeft.

Voor een traditionele tester is dit lastig omdat lang niet alle gebruikelijke documenten beschikbaar zijn. Als ze er wel zijn, zullen ze vaak niet tot in het door de traditionele tester gewenste detail uitgewerkt zijn. Maar zoals gesteld: documentatie is vaak een abstractie. Om de kennis te kunnen gebruiken moet er contact gezocht worden met de bron: de auteur. Een agile tester gaat dan ook niet op zoek naar documenten, maar naar mensen. Waar hij ten diepste naar op zoek is, is voldoende kennis om zijn testen op te baseren. Als die er niet in de vorm van documentatie is, zal deze er in de vorm van communicatie moeten zijn. Om goede testscripts te schrijven zal de ontwerper dus moeten participeren in het team – het liefst fulltime. Maar in ieder geval een aantal dagdelen per week om mee te werken met het ontwikkelen, testen en documenteren van het systeem. Daarmee wordt de testbasis dus niet een aantal documenten, maar de beschikbaarheid van de diverse noodzakelijke rollen in het projectteam.

2.6. TESTTECHNIEKEN

Agile projecten stellen andere eisen aan testtechnieken. Omdat een agile team flexibel is, samenwerkt, feedback vraagt en streeft naar eenvoud, moeten ze omwille van de efficiëntie van het testproces op een andere manier gebruikt worden.

Testen gebeurt in een agile project niet uitsluitend door de gespecialiseerde tester. Veel agilisten spreken over testen als rol: gedurende het project zal iedereen op enigerlei wijze bijdragen aan het testen. Tijdelijk ben je dan de tester, en een moment later weer ontwikkelaar of ontwerper. De ervaring leert dat dit niet zonder slag of stoot gaat, met name als het gaat om de vakinhoudelijke bagage. Veel ervaren ICT'ers missen eenvoudige technieken om consequent dezelfde kwaliteit van testen te leveren. In leertermen zijn ze 'onbewust onbekwaam'.

Echter, de stap van geen inhoudelijke testkennis naar meer kennis is geen gemakkelijke stap voor iemand die zich niet direct vakmatig bezig houdt met software testen. Testliteratuur is namelijk moeilijk te doorgronden voor niet-testers. Er zijn maar bijzonder weinig niet-testers die zich er toe kunnen zetten om testvakliteratuur uit te lezen. Zeker voor agile teamleden die weinig meer op hebben met het watervalmodel zal dit een brug te ver zijn. Daar moeten het testvak dus een oplossing voor vinden om ook agile teamleden aan het gestructureerd testen te krijgen.

Eenvoudige testtechnieken

Bij het voorbereiden van de cursus Testen voor Agile Teamleden vroeg ik mijn collega docent van de cursus Testtechnieken naar de uitwerking van de testtechniek Beslissingstabellen. Maar liefst veertien overvolle Powerpoint slides vertelden het verhaal hoe deze techniek correct toegepast moest worden. Dus inclusief alle mogelijke uitzonderingssituaties, mogelijke dekkingsgraad, etc. Dan heb je een dilemma: hoe vertel ik aan een groep mensen die zich niet primair met testen bezighouden, hoe ze deze techniek moeten toepassen? Mijn medevoorbereider zette mij direct met beide benen op de grond: “Maar een beslissingstabel bestaat toch in de basis maar uit vier stappen?” reageerde hij. Helemaal terecht!

Naast een betere toepasbaarheid van testtechnieken door agile teamleden is er nog een andere reden om kritisch naar de uitwerkingen van de testtechnieken te kijken. Dat is namelijk het feit dat deze testers verleiden tot uitgebreide, naar volledigheid neigende testscripts. Vaak leidt dit ook nog eens tot complexe uitwerkingen. Eenvoud en flexibiliteit, weet u nog?

Ook bij de toepassing van testtechnieken is er sprake van een dilemma. Traditionele testtechnieken zijn gericht op gespecialiseerde testers: ze zijn tot in detail beschreven, vaak vanuit een watervalinvalshoek, en neigen naar complexe uitwerkingen. Dit terwijl agile teams om veel communicatie vragen en snelle feedback nodig hebben. De aanpak moet dus makkelijk te communiceren én uit te voeren zijn.

De oplossing ligt in de beperking tot de essentie van de testtechnieken: daarmee wordt een goede basis gelegd en is er sprake van een gestructureerd testproces. De essentie van een testtechniek is vaak makkelijk uit te leggen, geeft snel inzicht in de mogelijke testgevallen en creëert daarmee makkelijk een gemeenschappelijk kader. Wij hebben dit uitgewerkt in zogenoemde Quick Reference Cards.

2.6.1. QUICK REFERENCE CARDS

Een Quick Reference Card (QRC) is een overzicht van een testtechniek op maximaal één vel A4. Hierop wordt weergegeven:

- De definitie van de testtechniek
- Het doel van de testtechniek
- Een voorbeelduitwerking
- Het principe van afleiden
- De basisaanpak

Definitie:	Het onderverdelen van mogelijke input- en outputvariabelen in verschillende klassen die logischerwijs hetzelfde systeemgedrag hebben		
Doel:	Het verhogen van de dekkingsgraad van input- en outputwaarden		
Principe:	Door testgevallen te baseren op equivalentieklassen in plaats van op elke mogelijke invoerwaarde, blijft het aantal testgevallen beperkt, terwijl toch een goede dekking verkregen wordt (elke waarde uit een klasse heeft dezelfde kans op het vinden van een fout).		
Aanpak:	stap 1. Bepaal de attributen: van welke gegevens moeten de inputwaarden worden vastgesteld? stap 2. Bepaal de attribuutklassen: welke waarden leiden tot andere gegevensverwerking/systeemgedrag? stap 3. Stel de testgevallen op: welke fysieke waarden ga je gebruiken?		
Voorbeeld:	(stap 1) **Attribuut**	(stap 2) **Geldige waarde**	(stap 3) **Ongeldige waarde**
	Maatschappij:	Kaart van aangesloten maatschappij	Diners Club, JCB
	Nummer:	Voldoet aan toetsingscriteria	1234567890
	Naam:	Geldige klantnaam	ABCDEFGHIJ
	Geldigheidsperiode:	Binnen termijn	12-2006

Voorbeeld van Equivalentieklassen

Deze aanpak heeft de volgende voordelen:

- Er wordt gewerkt vanuit de kern van de testtechniek, dus er is sprake van gestructureerd testen.
- Testgevallen worden sneller opgesteld, kunnen vervolgens sneller worden uitgevoerd en leiden tot snellere feedback.
- De uitwerking is door zijn eenvoud makkelijker te begrijpen voor niet-professionele testers.
- Kennis van gestructureerd testen is op deze manier voor teamleden en eindgebruikers makkelijker op te doen en vaardigheden worden sneller eigengemaakt.
- Wanneer de situatie daar om vraagt kan er altijd nog voor gekozen worden om meer traditioneel te werk te gaan, door de formele aanpak te kiezen.

Een dergelijke kernbenadering kan ook voor andere testtechnieken uitgewerkt worden, zoals:

- Grenswaardenanalyse
- Beslissingstabel
- Exploratory testing
- Proefgevallenanalyse
- Dataflowtest
- Procescyclustest
- Pairwise testing
- CRUD-matrix
- Profielen

2.7. AGILE TESTEN EN DSDM

De DSDM-aanpak is op dit moment de meest bekende agile aanpak in Nederland. De aanpak is ontstaan in de midden jaren negentig van de vorige eeuw, als uitwerking en vervolmaking van de RAD-methode (Rapid Application Development). Deze paragraaf beschrijft hoe agile testen zich verhoudt tot DSDM.

DSDM kent een negental principes, waarvan de voor het testen meest relevante er uit worden gelicht.

Iteratieve en incrementele ontwikkeling
Agile testen gaat uit van het frequent opleveren van software door het prioriteren als uitgangspunt voor de ontwikkelstrategie te nemen. Het ondersteunt incrementele ontwikkeling door er van uit te gaan dat wanneer iets eenmaal gebouwd is het nog kan wijzigen, met name in de testscripts en de wijzigbaarheid van de testscripts.

Actieve gebruikersinbreng
Omdat er iedere iteratie werkende software wordt opgeleverd, kan er ook iedere iteratie geaccepteerd worden. Door zowel de product owner als de eindgebruikers. Eindgebruikers kunnen echter ook binnen de iteratie, dus als onderdeel van het team, acceptatietests schrijven en daarmee de waarde voor de business en de inpasbaarheid van het systeem waarborgen.

Alle wijzigingen zijn terug te draaien
Agile testen gaat uit van eenvoud en van het nemen van beslissingen op een zo laat mogelijk moment. Door de korte feedback loops wordt er veel geleerd en is dus snel inzichtelijk wat de status is, en hoe er verbeterd kan worden.

Testen is geïntegreerd in de levenscyclus
Agile testers draaien vanaf de eerste dag mee in het team, zijn onderdeel van het projectteam. Er wordt gewerkt vanuit multifunctionele teams, die een gezamenlijke projectkamer hebben. Agile testen is ook vroeg testen, zodat fouten zo snel mogelijk geconstateerd worden.

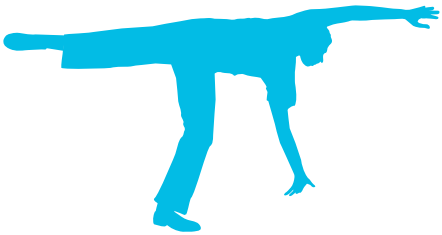
Samenwerkende en coöperatieve houding
Het team is verantwoordelijk voor de kwaliteit, niet de tester(s) alleen. Bij het voorbereiden en uitvoeren van tests wordt actief gestuurd op samenwerking. Belangrijke beslissingen, zoals de teststrategie en testtechnieken, worden altijd aan het team voorgelegd.

2.8. AGILE TESTEN EN RUP

Naast watervalgebaseerde methodes zijn veel bedrijven inmiddels overgestapt op de ontwikkelaanpak RUP® (Rational Unified Process). Deze aanpak wordt uitgegeven door IBM® en behelst de volledige ontwikkelcyclus van een softwareproduct. Het bestaat uit een viertal fases die geleidelijk in elkaar overlopen:

- De aanvangsfase ('Inception'), waarin er een visie gecreëerd wordt en de belangrijkste functionaliteit geïnventariseerd wordt. Dit resulteert in een software ontwikkelplan waarin onder meer de risico's, globale planning en kosteninschatting zijn opgenomen.
- De verduidelijkingsfase ('Elaboration'), waarin enkele use case specificaties worden uitgewerkt en een stabiele architectuur wordt ontwikkeld. Hier start het iteratieve werken. De ontwikkelomgeving wordt ingericht en eventueel wordt het software ontwikkelplan bijgewerkt met nieuwe inzichten. Er worden testontwerpen van de use cases opgesteld.
- De ontwikkelfase ('Construction'), waarin de use cases worden uitgewerkt, de code geschreven wordt en het testen plaatsvindt.
- De overgangsfase ('Transition'), waarin bugs worden hersteld, beheerders worden getraind en het systeem voldoen moet aan de acceptatiecriteria.

Alle fases worden ondersteund met templates en tools.



Doordat de fasering minder strikt loopt en er iteratief gewerkt wordt, kan de agile testen aanpak hier nauw bij aansluiten. Er kan al in een vroeg stadium getest worden en door de herhaling van de iteraties zullen eenvoud, feedback, samenwerken en flexibiliteit gevraagde eigenschappen zijn binnen het testproces.

De realiteit van veel RUP-projecten leert dat veel organisaties en projecten moeite hebben met het toepassen van het 'minder is meer' paradigma van agile. Het grote aantal templates dat met de methode meekomt, alsmede de detailwijze waarop het proces door RUP wordt beschreven zijn daar debet aan. Zo was er een organisatie die met een commissie van 'wijze mannen' vaststelde dat er dertien templates gebruikt moesten worden voor een doorsnee RUP-project. Het eerste doorsnee project leverde vervolgens 32 documenten op… Een duidelijke valkuil dus bij het toepassen van RUP wanneer er gestreefd wordt naar een agile ontwikkelaanpak.

3. DE IMPACT VAN AGILE

Zoals we in hoofdstuk 1 hebben kunnen lezen vraagt agile om een andere mindset, andere waarden in software ontwikkeling. In hoofdstuk 2 hebben we geconstateerd dat dit de nodige consequenties heeft voor de waarden in het testen en de practices die we daarin toepassen. Vervolgens kijken we in dit hoofdstuk welke impact dit heeft op onze huidige structuren. Dit is uitgewerkt op een drietal niveaus:

- Het niveau van de medewerker: de noodzakelijke competenties en vaardigheden die agile softwareontwikkeling en testen met zich meebrengt.
- Het niveau van de IT-organisatie: projecten, projectmanagement en portfoliomanagement.
- Het niveau van de business: geredeneerd vanuit generieke businessmodellen wordt bekeken welke mogelijkheden agile softwareontwikkeling biedt.

3.1. AGILE VOOR DE MEDEWERKER

Wanneer agile softwareontwikkeling en testen toegepast gaat worden binnen een organisatie, moeten de daarmee gepaard gaande andere normen en waarden geborgd worden. Het mag niet zo zijn dat het succes van een project afhangt van de toevallige samenstelling van een team. Dit is in een agile context net zo min gewenst als in een traditionele omgeving. Vandaar dat er aan gewerkt moet worden om mensen zich te laten ontwikkelen tot succesvolle agile teamleden, zowel door training en coaching als door het opdoen van praktische vaardigheden.

3.1.1. COMPETENTIES

Agile ontwikkelen en testen is bovenal samenwerken – als team met elkaar en met de klant. Dit is het belangrijkste uitgangspunt voor een succesvol agile team. Personen die van nature makkelijk communiceren zullen in een agile omgeving veel beter uit de verf komen, dan mensen die daar wat meer moeite mee hebben. Hier gaat het dan met name om de mondelinge communicatie – binnen de iteratie wordt er veel mondeling gecommuniceerd en afgestemd alvorens er een beslissing wordt genomen.

Omdat er binnen de iteratie multidisciplinair wordt gewerkt is inlevingsvermogen cruciaal: het gaat niet om het persoonlijk belang maar

"Making the simple complicated is commonplace; making the complicated simple, awesomely simple, that's creativity." – Charles Mingus

om het teambelang. Ieder teamlid dient in zijn eigen waarde gelaten te worden, de wederzijdse belangen dienen door iedereen onderkend te worden. Voor testers betekent dit ook dat ze meer op het teambelang gericht moeten zijn dan dat ze tot nu toe gewend zijn. Entry- en exitcriteria bijvoorbeeld dienen niet als een zwaard van Damocles boven de voortgang van het project te hangen, maar dienen zo smooth mogelijk gehaald te worden – dat is immers in ieders belang. Het sturen en coachen van het team op testen en kwaliteit is daarbij het uitgangspunt.

Als er één competentie essentieel is voor een tester in een agile project, dan is het wel de mate waarin hij of zij flexibel is: is diegene in staat zich makkelijk aan te passen aan wisselende omstandigheden? Dit wordt gevraagd omdat er tijdens de iteratie steeds sprake zal zijn van voortschrijdend inzicht. Over details wordt pas in een laat stadium besloten, namelijk in de iteratie. De uitwerking van testgevallen kan soms ook nog wijzigen doordat er voor een andere technische oplossing wordt gekozen. Een agile tester moet dus kunnen reageren op veranderingen.

Voor ieder agile teamlid is ook de mate van creativiteit van belang. Met creativiteit wordt bedoeld: de vaardigheid om zaken anders vorm te geven. Daarbij gaat het niet alleen om nieuwe concepten, maar ook om bestaande werkwijzen zodanig te kunnen vernieuwen dat ze meer aansluiten bij hun doelstelling. Creativiteit is altijd gebaseerd op kennis en ervaring. Dat zijn de bronnen waaruit je als creatieve specialist kunt putten. Inspiratie en motivatie zijn de drivers van creativiteit.

3.1.2. VAARDIGHEDEN

Omdat er zowel binnen de iteratie als over iteraties heen veel kan veranderen, streeft een agile teamlid ernaar om de zaken die hij ontwikkelt zo eenvoudig mogelijk te houden. Daardoor kunnen veranderingen ook daadwerkelijk snel doorgevoerd worden, en vormen de interne practices geen obstakel voor de business. Daarnaast bevorderen eenvoudige uitwerkingen de communicatie binnen teams. Het kunnen onderscheiden van hoofd- en bijzaken is daarbij een onmisbare vaardigheid.

Requirements in de iteratie zijn geprioriteerd. Dit betekent dat de scope niet in detail van te voren vaststaat. De opleverdatum, kwaliteit en het budget staan daarentegen wel vast. Dit mechanisme van 'belangrijke zaken eerst' beïnvloedt het volledige proces. Voor een teamlid is het dus van belang dat hij in staat is om zijn eigen werk ook te prioriteren. Een scherp beoordelings- en onderscheidingsvermogen over wat dan belangrijk is en in hoeverre het gewenst is om details uit te werken zijn zeer nadrukkelijk gevraagd.

Natuurlijk kun je al deze vaardigheden niet direct van een junior teamlid verlangen. Maar een junior medewerker kan best meedraaien in een agile team. Dan moet wel duidelijk zijn wat er van hem verwacht wordt op het gebied van vaardigheden en moeten er faciliteiten geboden worden om zich deze eigen te maken.

Wanneer we de verschillende competenties en vaardigheden bij elkaar trekken, wordt duidelijk dat agile werken specifieke kwaliteiten vergt. Kwaliteiten die op dit moment, in traditionele projecten, minder gevraagd worden omdat ze minder van belang zijn om het model te laten slagen. Voor een heel aantal teamleden zal het gaan om al dan niet 'verborgen' kwaliteiten. Geef mensen de tijd om deze te ontwikkelen binnen de nieuwe context.

Maar de ervaring leert ook dat een aantal van de medewerkers de gewenste kwaliteiten onvoldoende bezitten en zich onvoldoende kunnen ontwikkelen om effectief te kunnen zijn binnen een agile project. Voor hen zullen dus nieuwe, andere uitdagingen gecreëerd moeten worden.

3.2. AGILE VOOR DE IT-ORGANISATIE

De kracht van agile ontwikkelen ligt in flexibiliteit. Daarin ligt ook de relatieve zwakte. Tenminste, als het onvoorspelbare wordt gezien als een zwakte. Door de iteratieve werkwijze en de maandelijkse nieuwe cyclus aan requirements is het project minder voorspelbaar als het gaat om scope en details. Het exacte resultaat, traditioneel vaak vooraf gedefinieerd in termen van scope, staat namelijk niet vast. Maar daar komen andere zekerheden voor terug:

- Vaste oplevermomenten die nooit gemist worden – het mechanisme van prioriteren en timeboxen zorgt er namelijk voor dat minder belangrijke functionaliteit ondergeschikt blijft aan tijd.
- Budgetten blijven constanter: er wordt veelal gewerkt vanuit een bezettingsgraad van medewerkers per iteratie, een percentage effectieve uren en het aantal (naar rato) te besteden uren op menscapaciteit. Er kan dus wel geschakeld worden door meer mensen in een iteratie te laten meewerken, maar ook hier kan simpelweg geprioriteerd worden.

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius -and a lot of courage- to move in the opposite direction." – Albert Einstein

- De kwaliteit is constant als er sprake is van een agile testen aanpak: door de integrale aanpak waarbij er tijdens de iteratie op alle niveaus getest wordt, is de kwaliteit van het opleverproduct altijd in orde. In een agile project wordt de kwaliteit niet ondergeschikt gemaakt aan de tijdigheid. Er worden geen extra features tijdens de iteratie toegevoegd of gewijzigd en het team werkt volgens een afgesproken kwaliteitsniveau.
- Waarde voor de business: een agile team is gericht op samenwerking en dat betekent dat ook stakeholders nauw betrokken worden in het proces. De voortgang is altijd transparant. En wanneer het bijvoorbeeld gewenst is dat iemand uit de beheerafdeling kennis over het systeem gaat opdoen, dan kan die persoon meedraaien in de iteratie. De traditionele kloof tussen project en beheer hoeft er dus niet te zijn.

Vanuit de IT-organisatie zal er op een andere manier gestuurd moeten worden dan op functionaliteit. Het paradigma is niet 'is het systeem compleet zoals gedefinieerd in het ontwerp', maar het wordt 'bevat het ontwikkelde systeem voldoende waarde om gereleast te worden'. Dit heeft overigens wel als consequentie dat er op programmaniveau goed gekeken moet worden naar de meerwaarde die de projecten opleveren. Er zal nauwer tussen de projectmanagers onderling afgestemd moeten worden en de klant die meerdere projecten heeft lopen zal kritisch naar zijn eigen wensen moeten kijken.

De projectorganisatie van een agile project is per definitie van invloed op de gebruikersorganisatie van de opdrachtgever: er wordt van uitgegaan dat zij een actieve en betrokken rol speelt in het ontwikkelproces. Gebruikers zijn vaak een integraal onderdeel van het projectteam. Veel van de klantrol zoals besproken in hoofdstuk 2 wordt gezamenlijk met de product owner ingevuld. Dat betekent dus dat voor ieder project er eindgebruikers uit hun eigen organisatie losgeweekt moeten worden om die rol te kunnen vervullen. Uiteindelijk wordt er daardoor betere functionaliteit ontwikkeld, is deze meer *fit for purpose*, en is de acceptatiegraad van het nieuwe systeem.

3.3. AGILE EN DE BUSINESS

Agile beoogt nogal wat: snellere time to market en return on investment, het leveren van veel waarde voor de business en continue kwaliteit… Kan dat allemaal wel? Zoals het een goed paradigma betreft kun je met agile software ontwikkelen meerdere kanten op. Voor ieder bedrijf levert het weer andere voordelen op.

Aan de hand van de drie waardedisciplines van Treacy & Wiersema worden de mogelijkheden van agile softwareontwikkeling benoemd. De practices uit agile software

ontwikkelen ondersteunen de waardedisciplines op verschillende manieren. Daarmee is agile een uitstekend ICT-framework. Het ondersteunt ook bedrijfsstrategieën.

3.3.1. OPERATIONAL EXCELLENCE

Wanneer een bedrijf streeft naar een zo laag mogelijke kostprijs van een product, een betrouwbare uitstraling wil hebben en zelden te betrappen is op het maken van fouten, dan spreken we van Operational Excellence. Dit wordt vaak vertaald met kostenleiderschap – het leveren van een uitstekende combinatie van prijs, kwaliteit en koopgemak.

Doordat een bedrijf streeft naar zo laag mogelijke kosten, zal het snel return on investment willen zien.

Door de iteratieve werkwijze kan iedere maand werkende software worden opgeleverd, waardoor de initiële investeringen in principe beperkt zijn, terwijl het systeem al heel snel in productie kan. Daarbij streeft het bedrijf ook naar zo weinig mogelijk fouten. Vrijwel iedere ICT'er kent de curve van Boehm: fouten oplossen wordt steeds duurder naarmate ze later in het traject gevonden worden. Doordat agile ontwikkelen uitgaat van een iteratie waarin ontwerp, bouw, test en oplevering in een integrale aanpak wordt doorlopen, zijn fouten snel gevonden. Ook fouten die voorheen laat in de ontwikkelcyclus zaten. Binnen de iteratie worden alle noodzakelijke tests uitgevoerd, tot en met de acceptatietest aan toe. Door het intensieve testproces worden veel fouten voorkomen, en in ieder geval snel ontdekt.

Inherent aan kostenleiderschap is het afwegen van kosten versus baten. Dat wordt deels al ondervangen door de snellere return on investment. Maar ook het prioriteren van de requirements helpt hierbij: de product owner moet afwegen of requirements echt belangrijk zijn en of de investering in het ontwikkelen ervan zich op de lange termijn gaat uitbetalen.

3.3.2. CUSTOMER INTIMACY

Customer Intimacy is vooral het hoog in het vaandel hebben van de klantrelatie: het bieden van uitnemende service, het snel kunnen reageren op wensen van de klant en het bieden van maatwerkoplossingen – oftewel klantenpartnerschap. De klant staat centraal in het volledige bedrijfsproces.

Agile is zeer geschikt voor het uitdrukken van een partnership met de klant: wanneer de product owner daadwerkelijk de belangen van één klant behartigt en er ontwikkeld wordt op basis van één team voor één klant dan is het wellicht de enig juiste oplossing. Agile biedt in deze context de volgende voordelen:

- Er wordt een organisatorisch framework gedefinieerd, waarin de klant kan vragen en het team kan leveren.
- De vraagwens kan op zeer korte termijn worden opgeleverd: in maandelijkse iteraties, in de vorm van werkende software.
- Door de eendimensionale rol van de product owner kan de te ontwikkelen functionaliteit volledig fit for purpose voor de gewenste oplossing zijn – klantgerichtheid avant la lettre.
- Doordat het team samenwerkt met de product owner en daarmee kennis krijgt van de klant en zijn proces, zullen er ook door de teamleden betere besluiten genomen kunnen worden.

De product owner speelt hierin nadrukkelijk een sleutelrol: hij zal de klantwens zeer goed moeten kennen en ook weten welke strategie er achter de wens ligt. Daarbij moet hij dit kunnen uitdragen naar het ontwikkelteam toe.

3.3.3. PRODUCT LEADERSHIP

Bij product leadership, oftewel productleiderschap, ziet de klant het bedrijf als innovatief en creatief, en is bereid daar meer voor te betalen. Een product leader is de concurrentie steeds een slag voor omdat hij in een blijvende behoefte voorziet. Het bedrijf zelf is vooral productgericht. De focus ligt op het produceren van een constante stroom van nieuwe producten.

Vanuit deze waardediscipline biedt agile software ontwikkelen diverse mogelijkheden.

- Een agile ontwikkelproces is in staat om snel tot werkende software te komen. Producten kunnen dus snel ontwikkeld worden en door de korte feedbackloop kan er snel besloten worden of het product de goede kant op evolueert.
- De product owner bepaalt de prioriteiten van de requirements, hij is dus degene die weet wat belangrijk is. In een agile ontwikkelproces wordt er uitgegaan van één product owner. Er kan sprake zijn van een vertegenwoordiger die de belangen van meerdere stakeholders behartigt. Hij is er verantwoordelijk voor dat de requirements

die ontwikkeld worden aan de marktbehoefte voldoen. Doordat er sprake is van één rol en in principe één persoon wordt er sneller besloten en kan het ontwikkelteam sneller acteren op gewenste marktveranderingen. Op die manier kan er een product opgeleverd worden dat optimaal is afgestemd op de marktbehoefte.

- Veel organisaties die overstappen op agile softwareontwikkeling melden toegenomen productiviteit van hun IT-projecten[13], blijkt uit onderzoek van VersioneOne: 87 procent van de ondervraagden meldde meer dan 10 procent toegenomen productiviteit en 55 procent meldde een stijging van 25 procent of meer. In een bedrijf waar de focus ligt op het produceren van nieuwe producten, betekent dit dat er met hetzelfde aantal resources méér geproduceerd kan worden. Daarmee kan er nog meer concurrentievoordeel behaald worden.

3.4. AGILE ALS ONDERSTEUNENDE STRATEGIE

Agile ontwikkelen en testen biedt op basis van voorgaande inzichten een uitstekende basis voor een uitgekiende bedrijfsstrategie. Een strategie waarbij een businessgerichte inrichting van het softwareontwikkelproces als een driver wordt gebruikt om de ambities te verwezenlijken. Ongeacht of die ambities op het niveau van kostenverlaging, productleiderschap of klantpartnerschap zijn. Voor ieder van de waardedisciplines heeft een agile ontwikkelaanpak practices die de businessstrategie ondersteunen. De korte iteraties worden gebruikt als feedback voor beslissingen, maar ook om werkende software op te leveren en daarmee een snellere return on investment te faciliteren. De product owner is de spin in het web als het gaat om het prioriteren van het werk. Daarmee kan er meer waarde gecreëerd worden voor de klant, of voor de interne organisatie wanneer er sprake is van eigen productontwikkeling. Het in de ontwikkelcyclus geïntegreerde testen biedt continu inzicht in de voortgang en kwaliteit van hetgeen het projectteam aan het ontwikkelen is.

Het is dus niet langer zo, dat terwijl een ICT-systeem ontwikkeld wordt de bedrijfsdoelstellingen bevroren moeten worden. Agile biedt mogelijkheden om wijzigende eisen en wensen te faciliteren. Agile testen biedt een aanpak die hier naadloos op aansluit. Voor de business is dit een mogelijkheid die gegrepen zal worden, voor testers is het de uitdaging dit waar te maken.

NAWOORD

Van alle vakdisciplines staat ICT misschien wel als enige midden op het kruispunt van technologie en economie. Er wordt voortdurend gesproken over de belangrijke vernieuwingskracht die ICT biedt om processen beter en sneller te organiseren. Meer variëteit, meer snelheid, meer maatwerk in producten en diensten, hogere kwaliteit en tijdsonafhankelijkheid worden al snel als zegeningen genoemd die ICT organisaties kan verschaffen. Niet zelden ontstaan op die manier pretenties die eerder verlammen dan versnellen. Dat lot heeft de ICT langere tijd boven het hoofd gehangen. Pretenties waren er maar realisatiekracht schoot bij herhaling tekort. Niet omdat de ICT dat niet kon, maar veelal omdat de afstand tussen ICT en de te innoveren business te groot bleek. ICT bleef te veel een geïsoleerde technische kolomactiviteit van specialisten.

Pas de laatste drie à vijf jaar hebben we op dat punt een echte doorbraak bereikt. ICT komt ook op de tafel van het lijnmanagement terecht om van daaruit werkelijke effecten te sorteren. In het licht van deze ontwikkeling zijn de afgelopen jaren ook vele delen van de ICT van karakter veranderd: geprofessionaliseerd en gemoderniseerd. Met name de disciplines aan de voorkant van de ICT, zoals proces en architectuur hebben onder invloed van de business een enorme vlucht genomen. Interessant is dat de specialismen die meer aan de achterkant van de ICT zitten nog relatief weinig van deze ontwikkeling profiteerden, met als markant sluitstuk het testvak. Te vaak is gedacht dat je alles uit de ICT wel kunt moderniseren en verbeteren maar dat het testen een relatief statische en onwrikbare gebeurtenis van het eind van de systeemontwikkeling was.

Het is de verdienste van het verhaal van Anko Tijman dat hij laat zien dat alles wat het ICT-vak modern en aan de eisen van de tijd aangepast heeft gemaakt ook zal gaan gelden voor het testen. En dat er methoden en technieken van testen zijn die volstrekt afrekenen met alle vooroordelen die over dat deel van het vakgebied gelden.

Testen kan interactief en iteratief en daarmee in communicatie met hen die van de geteste systemen gaan profiteren. Daarmee wordt niet alleen belangrijke kwaliteitswinst geboekt, maar wordt ook gezorgd dat in een vroeg stadium gebruikers op alle niveaus meer betrokken worden bij de implicaties in werk-

procesverandering die ICT-systemen nu eenmaal altijd met zich meebrengen. Toepassing van deze inzichten maakt dan ook dat de impactanalyse van systeemontwikkeling niet zoals in de klassieke benadering pas plaatsvindt na volledige afronding van het systeem, maar in de interactieve testperiode. Daarmee wordt de testdiscipline in de ICT een belangrijke bouwsteen. En zo draagt testen bij aan de versnelde effectuering van bedrijfsmodernisering. De methodische aanpak zoals verwoord in dit boek kan daarbij als boeiende en nuttige handreiking gezien worden en nodigt uit tot navolging.

Dr. Tom Rodrigues
Raad van Bestuur Ordina
December 2006

LITERATUUR

Lessons Learned in Software Testing
Kaner, Bach, Pettichord
Wiley, 2001

Testing eXtreme Programming
Crispin, House
Addison-Wesley, 2002

Lean Software Development:
An Agile Toolkit for Software Development Managers
Poppendieck, Poppendieck
Addison-Wesley, 2003

Extreme Programming Explained – Embrace Change
Beck, Andres
Addison-Wesley, 2004

Agile Software Development with Scrum
Schwaber, Needle
Prentice Hall, 2002

Agile Software Development
Cockburn
Addison-Wesley, 2001

Software Test Automation
Graham, Fewster
Addison-Wesley, 1999

WEBSITES

http://www.agilealliance.org

http://www.agilemanifesto.org

http://www.agile2007.org

http://www.controlchaos.com

http://www.dsdm.nl

http://www.methodsandtools.com

http://www.scrumalliance.org

http://www.stickyminds.com

http://www.testing.com

http://www.testforum.nl

http://www.testinglessons.com

http://www.xprogramming.com

http://tech.groups.yahoo.com/group/agile-testing/

http://tech.groups.yahoo.com/group/xp-nl/

EINDNOTEN

1. Testen volgens TMap®, 2e druk, Pol, van Veenendaal, Teunissen, Uitgeverij Tutein Nolthenius, 1999

2. Tmap Next, Koomen, van der Aalst, Broekman, Vroon, Uitgeverij Tutein Nolthenius, 2006

3. Testing Computer Software 2nd edition, Kaner, Falk, Nguyen, Wiley, 1999

4. Corporate IT Leads The Second Wave of Agile Adoption - Forrester Research, nov 2005

5. Extreme Programming Explained: Embrace Change, Beck, Addison-Wesley, 1999

6. Extreme Programming Explained: Embrace Change, Beck, Andres, Addison-Wesley, 2004

7. Meer over de aanpak van Toyota en agile software development is te vinden in 'Lean Software Development: An agile toolkit for software development managers' door Mary en Tom Poppendieck, Addison Wesley, 2003

8. Boehm, 'Software Engineering Economics', Prentice Hall, 1979

9. Sander Hoogendoorn, 'Smart, succesvol samenwerken in systeemontwikkeling', Ordina, 2003

10. Scrum is een agile methode. Het is geen acronym, maar Scrum staat voor de herstart van het rugby spel, waarbij de teams met de schouders tegen elkaar staan en de bal in de groep wordt gerold.

11. Voor meer informatie over de agile projectmanagementmethode Scrum zie www.controlchaos.com

12. Een waardevol boek rondom het professioneel opzetten van geautomatiseerd testen, hoewel niet specifiek agile, is 'Software Test Automation' van Fewster & Graham, Addison-Wesley, 1999

13. Bron: State of Agile Development Survey, www.VersionOne.com (2006)