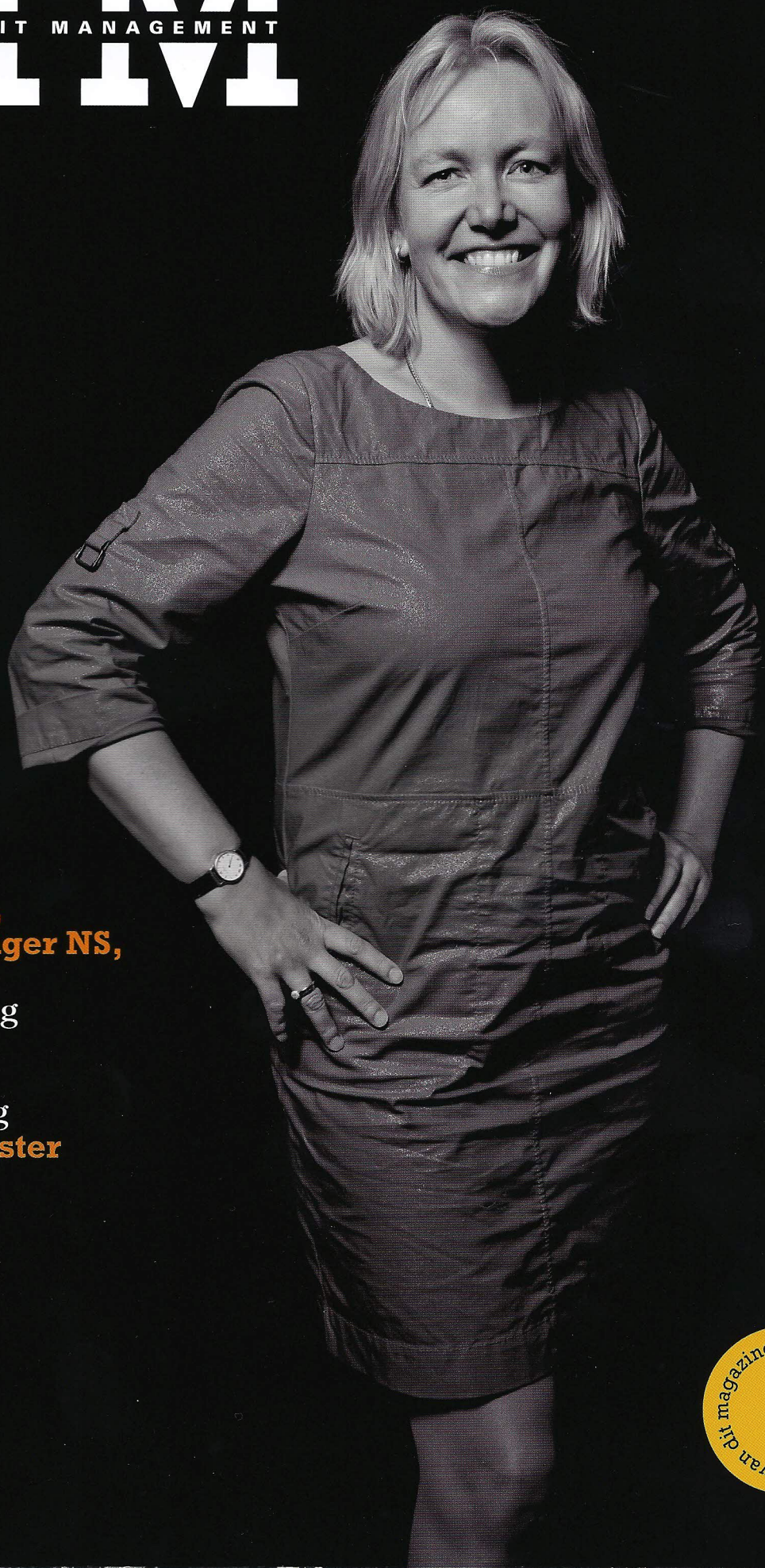
DECEMBER 2013 JAARGANG 6 INNOVATIE – INFORMATIE - APPLICATIES

TITM

TIJDSCHRIFT IT MANAGEMENT

6 Informationtrends



Jantina Woudstra, programmamanager NS, over innovaties en kwaliteitsverbetering

De wederopstanding **van de softwaretester**

De weg naar intelligente processen

Zie ook andere zijde van dit magazine!

Meeveranderen biedt nieuwe kansen naar een volgende carrièrestap

De wederopstanding van de softwaretester

De toekomst van de softwaretester lijkt erg onzeker. Nieuwe tooling, agile werken en multidisciplinair denken en handelen zijn ontwikkelingen die doen vermoeden dat het over en uit is met de tester. Maar niets is minder waar. De softwaretester is bezig met een wederopstand. Er zijn vijf redenen om deze conclusie te trekken.

Bij de traditionele testmethoden worden nogal wat rijtjes met fasen, activiteiten en producten opgesomd en doorgeakkerd. Een tester die zo'n methode klakkeloos overneemt, richt een veel te duur en tijdrovend testproces in en prijst zichzelf uit de markt. Niet meer doen dus. Neem niet alles over maar pas zaken adaptief aan in de context waarin je werkt. Eén gecombineerde activiteit is echter altijd een absolute must: de uitvoer van een productrisicoanalyse en het opstellen van een teststrategie met alle betrokkenen. Anders gezegd, samen met de betrokkenen moet een goede balans worden gevonden tussen de af te dekken risico's enerzijds en de daarmee gemoeide kosten en tijd anderzijds. Daar mag het bij blijven. Tegelijkertijd kun je uiteraard samen met de projectbetrokkenen bepalen wat nog meer nodig is. Dat is een eenvoudige legpuzzel. Je gebruikt, wijzigt of voegt alleen die stukjes toe die in jouw context passen. Ongeacht de methode (TMap, ISTQB, Testframe of ISO29119) die je gebruikt. Een vuistregel bij het kiezen is, is het stukje voor mij als tester nodig om mijn werk te kunnen doen, of is het stukje voor de stakeholders nodig? Is het antwoord op beide vragen nee, dan kun je het stukje simpelweg snel dumpen.

Katalyseer kwaliteitsverbeteringen

Met de inzet van een agile ontwikkelaanpak, zoals het veelvuldig gebruikte scrum, heeft ieder teamlid een testrol. Dus ontwerpers, programmeurs, beheerders en gebruikers testen allemaal. Daarmee is het plausibel te stellen dat testers overbodig zijn. Kort door de bocht? Het is de praktijk van vandaag. Zo gaf een IT-manager van een grote organisatie onlangs aan dat de testafdeling gehalveerd kon worden nu er met Scrum gewerkt wordt. Deze IT-manager moet weten dat je niet kan verwachten dat alle teamleden opeens over dezelfde uitgebreide testvaardigheden beschikken als de ervaren tester. Draai het eens om. Een scrumteam met alleen testers. Kunnen die dan 'opeens' allemaal ontwerpen en programmeren? Nee toch? In de praktijk moeten testers hun teamleden dus ook helpen om hun rol als tester in een Scrum-aanpak zo goed mogelijk te kunnen vervullen. Bijvoorbeeld door hulp te bieden bij het opstellen van kwalitatief goede *userstories*. De tester kan de ontwerper of gebruiker toetstechnieken bijbrengen (bijvoorbeeld het INVEST-model), of de programmeur helpen bij het opstellen van unittesten, en de gebruiker helpen bij de acceptatietest. Tenslotte moet de tester modereren bij de eerder genoemde productrisicoanalyse. En zo beweegt de testprofessional veel meer richting katalysator van kwaliteitsverbeteringen.

> De IT-manager gaf aan dat door Scrum de testafdeling gehalveerd kon worden

Risicobeperkende maatregelen

In bepaalde omgevingen, zoals bij de ontwikkeling van mobiele apps en websites, waar het motto 'eerst snelheid dan kwaliteit' opgeld doet, wordt software vaak zo snel mogelijk in productie ge-

nomen. Op zo'n moment wordt de tester volledig gepasseerd. Liever worden eventuele fouten tijdens het productieproces opgelost, dan de applicatie vooraf te testen. Dat kost te veel geld en tijd. Door de korte iteraties en vrijwel continue inproductienamemogelijkheden, kost het waarschijnlijk ook echt minder tijd en geld. Echter, waar sprake is van serieuze risico's, zoals risico's voor de samenleving, reputatieschade of grote impact op bedrijfsresultaten, is testen van cruciaal belang. Uiteindelijk gaat het om de balans tussen wat het kost om de fout tijdens het ontwikkeltraject op te sporen en de kosten van fouten tijdens het productieproces. Het is van belang dat de tester dichter tegen de business aan zit om hier inzicht in te krijgen. Daarmee kan hij of zij een goed advies geven over de te volgen teststrategie; voornamelijk over het mogelijke afdekken van eventuele risico's. Denk daarbij ook aan andere risicobeperkende maatregelen dan testen alleen.

Requirements-elicitatie en acceptatietest

Er worden steeds meer tools voor modeldriven-development ontwikkeld die op termijn – foutloze – code moeten genereren op basis van requirements of functionele specificaties. Hiermee lijkt de rol van de tester wederom uitgespeeld. Een tester moet dan opschuiven richting de klant ofwel gebruiker, en hem ondersteuning geven bij zowel het eliciteren, als bij het SMART maken van de requirements door bijvoorbeeld toetstechnieken te introduceren en te gebruiken. Populair gezegd, de tester moet van rechts naar links en naar boven in het V-model opschuiven. Een aanpak als PointZERO kan hierbij goede ondersteuning geven. Bovendien, al zou de code uit requirements gegenereerd worden, en ook al zou dit een foutloze code zijn, dan blijven acceptatietests nog steeds noodzakelijk. Een foutloze code is immers geen garantie dat deze applicatie het werk van bijvoorbeeld de eindgebruiker op de gewenste wijze ondersteunt. De productieacceptatie- en ketentests blijven ook nog steeds nodig. Hierbij zijn immers andere applicaties betrokken, dan die opgeleverd zijn. Een waarschuwing is hier wel op zijn plaats, overdrijf het niet met te veel verschillende testsoorten.

Geen hangmat maar een vangnet

Willem Drees heeft ooit gezegd: "De bijstandswet moet een vangnet zijn, geen hangmat". Dat geldt ook voor softwaretesten. In veel organisaties of projecten wordt een grote verscheidenheid aan testsoorten gebruikt. Vaak in reeksen achter elkaar uitgevoerd. Je kunt het zo gek niet bedenken. Een reeks als de unittest, unitintegratietest, systeemtest, systeemintegratietest, functionele acceptatietest, gebruikersacceptatietest, ketentest en productieacceptatietest is bijvoorbeeld geen uitzondering. Zo wordt de indruk gewekt dat ontwikkelaars hier misdan wel gebruik van maken. "Ik lever de code op tijd op en dan hoor ik wel of er fouten in zitten, want na mij wordt er toch nog uitgebreid getest." En hop, de ontwikkelaar duikt in zijn hangmat! Opvallend is dat in organisaties of projecten waar het ongebruikelijk is zoveel testsoorten in te richten en uit te voeren, software met minder fouten wordt opgeleverd. Hoe kan dat? Het antwoord is eigenlijk heel simpel. Door de vele tests voelt de programmeur zich minder verantwoordelijk voor de kwaliteit van zijn product. In situaties waar de software vrijwel direct na een unittest in productie wordt genomen, móet de programmeur wel zijn verantwoordelijkheid nemen. Als het fout gaat, weet tenslotte iedereen wie daarvoor verantwoordelijk is. Kortom, het verminderen of samenvoegen van testsoorten, en de ontwikkelaar, of eigenlijk alle projectleden, meer aanspreken op hun verantwoordelijkheid bevordert het opleveren van kwalitatief goede software. Zo wordt testen weer een vangnet in plaats van hangmat.

> Foutloze code is geen garantie dat de gebruiker op de gewenste wijze wordt ondersteund

Tot slot

De conclusie is simpel; alles verandert. Niet alleen het IT-vak, maar ook het testvak, en dat betekent dat de tester mee moet veranderen. Niets doen betekent het einde van de loopbaan in softwaretesten. Daarentegen geeft meeveranderen enorm veel nieuwe kansen naar een volgende carrièrestap. De wederopstand van de softwaretester valt of staat met verdieping en verbreding van de functie. De nieuwe generatie testers verschuift op deze manier van laatste man naar verdedigende middenvelder. Een ware wederopstand dus.

Leo van Aalst is senior consultant bij Sogeti, auteur van testboeken en lector software quality and testing aan de Fontys Hogescholen