http://www.computable.nl/artikel/opinie/development/4875709/1277180/vijf-redenen-waarom-testen-dood-is.html

Vijf redenen waarom testen dood is

19-09-2013 10:21 | Door Leo van Aalst | Lees meer artikelen over: Testing, Agile, Apps, Scrum, TMap, ISTQB, TestFrame | Er zijn 11 reacties op dit artikel | Permalink

Met name in de afgelopen drie jaar is er eindeloos veel geschreven, gesproken en gedaan over de toekomst van software testen. Een rol die aan het einde van zijn latijn zou zijn. Hieronder vindt u vijf redenen waarom testen dood is. Argumenten die wellicht houtsnijden. En tegelijkertijd weinig overtuigend zijn.

Het gebruik van gestructureerde testmethoden kosten meer tijd en geld

Bij de traditionele testmethoden worden nogal wat rijtjes met fasen, activiteiten en producten opgesomd en doorgeakkerd. Als de tester dat proces volgt, gaat dat inderdaad veel tijd en geld kosten. Niet doen dus. Neem niet alles klakkeloos over maar pas zaken aan in de context waarin je werkt. Eén activiteit is altijd een absolute must. De uitvoer van een productrisicoanalyse. En verder bepaal je samen met projectbetrokkenen wat nog meer nodig is. Op basis van de geïdentificeerde risico's bekijk je welke activiteiten, technieken, producten en personen een rol spelen om die risico's af te dekken. Ik zie dit als 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. De tester moet in staat zijn op adaptieve wijze een testaanpak in een bepaalde context te gebruiken. Anders gezegd, hij of zij moet een goede balans vinden tussen de af te dekken risico's enerzijds en de daarmee gemoeide kosten en tijd anderzijds. Daarbij moet hij/zij een goede balans vinden tussen de af te dekken risico's en de daarmee gemoeide kosten en tijd centraal staan.

De inzet van nieuwe ontwikkelaanpakken maakt testen overbodig

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. Dus daarmee is het plausibel te stellen dat testers overbodig zijn. Echter, twee kanttekening zijn hierbij cruciaal. Bij gedeelde verantwoordelijkheid is niemand verantwoordelijk, wordt wel gezegd. Dat is hier ook van toepassing. Als iedereen als de tester wordt beschouwd, is eigenlijk niemand de tester! En verder mag je niet verwachten dat alle teamleden opeens over dezelfde uitgebreide testvaardigheden beschikken als de ervaren tester. Dat betekent dat testers hun teamleden moeten helpen om deze rol in een Scrum-aanpak zo goed mogelijk vervullen. Bijvoorbeeld hulp bieden bij het opstellen van kwalitatief goede user stories. De tester kan daarbij de ontwerper of gebruiker toetstechnieken bijbrengen. 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, is mijn stellige overtuiging, moet de testprofessional bewegen richting katalysator van kwaliteitsverbeteringen.

Testen is duurder dan het verhelpen van productieverstoringen

Testen kost veel geld en tijd. Soms is het beter de software zo snel mogelijk in productie te nemen. En daarmee eventuele fouten tijdens het productieproces op te lossen. Door de korte iteraties en vrijwel continue inproductiename-mogelijkheden die er zijn, kost dat immers minder tijd en geld. Eerst snelheid dan kwaliteit, is dan het motto. Dat kan ook zeker in een aantal omgeving zoals de inzet van mobile apps en websites. Echter, waar sprake is van serieuze risico's is testen van cruciaal belang. Risico's voor onze samenleving, reputatieschades, of grote impact op bedrijfsresultaten. 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 gaat zitten om hier inzicht in te krijgen. Daarmee kan hij of zij een goed advies geven over de te volgen teststrategie. Met name ook hoe eventuele risico's kunnen worden afgedekt. Denk daarbij ook aan andere risicobeperkende maatregelen dan testen alleen!

Tooling zorgt ervoor, dat ontwikkelaars geen fouten meer maken

Al 25 jaar lang wordt geroepen dat testen voorbij is zodra tools voorhanden zijn die foutloze codes genereren op basis van functionele specificaties. Het bevestigt eens te meer hoe slecht we zijn in het voorspellen van de it-toekomst. We komen inderdaad steeds verder met de tooling die voorhanden is, en waar nog volop aan gewerkt wordt. Toch is dat helaas geen gemeengoed. En ook al zou de code uit requirements gegenereerd worden, en ook al zou dit foutloze code zijn, dan blijven acceptatietests nog noodzakelijk. Foutloze code is nog steeds geen garantie dat deze applicatie het werk van bijvoorbeeld de eindgebruiker op de gewenste wijze ondersteunt. Een tester kan ondersteuning geven bij het opstellen van de requirements, bijvoorbeeld door toetstechnieken te gebruiken, en bij het opstellen van de acceptatietests. Tegelijkertijd blijven de productieacceptatie- en ketentests ook nog steeds nodig.

Meer Opinie

02-10 Whitelisting is klaar voor enterprise endpoints
02-10 Verzeker je start-up in een paar eenvoudige...
02-10 Enterprise IT = Ctrl Alt Delete 14
01-10 BYOD 2.0 is mobile device management voorbij
01-10 Dalende ICT-kosten gemeenten zeggen niet alles 1
01-10 Smartphones en biometrie vormen het grote plaatje 21
30-09 Virtualisatie verandert ook de telefonie 12
30-09 Naast mensen ook spullen nodig voor beveiliging 3
30-09 Softwareontwikkeling lijkt wel op vechtkunst 7
27-09 Technisch applicatiebeheer kan leren van Fyra 6

Meer Development

02-10 Overheid en TU Delft investeren in quantum
02-10 Enterprise IT = Ctrl Alt Delete 14
02-10 Triggre start modelgedreven ontwikkelplatform 1
30-09 Atos beheert applicaties GBO Provincies en IPO
30-09 Freelance webdevelopers starten Dutch Web Alliance
30-09 Softwareontwikkeling lijkt wel op vechtkunst 7
27-09 Technisch applicatiebeheer kan leren van Fyra 6
27-09 ICT-afdeling blijft rots in de branding 5
26-09 Wil de echte expert opstaan? 25

Gerelateerde artikelen

26-09-13 Vijf competenties voor de nieuwe Agile-testers 2
24-09-13 MBT: revolutie of oude wijn in nieuwe zakken
17-09-13 Mobiel testen is een wereld van verschil 4

In Nederland wordt een grote verscheidenheid van testsoorten gebruikt. Vaak in reeksen achter elkaar uitgevoerd. Soms krijg ik de indruk dat ontwikkelaars hier mis- dan 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." Het valt op dat in landen waar het niet gebruikelijk is zoveel testsoorten in te richten en uit te voeren software met mindere 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, moet de programmeur wel zijn verantwoordelijkheid nemen. Als het fout gaat, weet tenslotte iedereen wie daarvoor verantwoordelijk is. Kortom, ik pleit voor het verminderen en/of samenvoegen van testsoorten en de programmeur meer verantwoordelijkheid geven. Dat vergroot de kwaliteit van de software.

Alles wordt al in de cloud getest

Laat de beta tests op de applicatie in de cloud uitvoeren en alle fouten worden gevonden. Voor de duidelijkheid, niet alle applicaties zijn hiervoor geschikt. Op basis van strategische redenen kiezen organisaties ervoor niet alle applicaties in de cloud te zetten. Maar inderdaad, beta testers vinden veel fouten. Dit zijn veelal de triviale en niet de ingewikkelde fouten waar je toch echt een testscenario voor moet schrijven. Daarbij wordt geen rekening gehouden met eventuele risico's. Een ongeorganiseerde groep mensen vuurt een schot testhagel af op de applicatie.

Onlangs hoorde ik van een medewerker van een 'game design'-bedrijf dat er nog een ander nadeel aan zit. 'De beta testers', zei hij, 'willen het spel spelen en rapporteren vaak alleen die fouten die hen verhinderen het spel verder te kunnen spelen. Dus rapporteren ze niet alle fouten.' Nou zou het natuurlijk niet verstandig zijn de kracht van de cloud onbenut te laten. Maar in plaats van een applicatie zomaar in de cloud te laten testen, gaat mijn voorkeur uit naar crowd-sourced testing. Dus stel de applicatie aan bepaalde zorgvuldig geselecteerde mensen ter beschikking om te testen. Of nog beter, expert-sourced testing. Biedt de applicatie aan mensen met testexpertise en/of kennis van de applicatie om te testen.

Leo van Aalst, senior consultant, auteur en lector software kwaliteitszorg Sogeti

Meer weten over de toekomst van testen? Luister naar de keynote speech van Leo van der Aalst op de 'tmapdag', de grootste testingconferentie van ons land op 1 oktober 2013 in Bussum.

Over de auteur

Leo van der Aalst heeft ruim 25 jaar ervaring in de it. Na zijn carrièrepad van programmeur tot programmamanager, heeft hij zich gespecialiseerd in het testvak.

Van der Aalst is co-auteur van de TMap NEXT, TMap NEXT BDTM en TMap NEXT in Scrum boeken en ontwikkelde onder andere diensten voor Agile-testen, de inrichting van testorganisaties en voor testoutsourcing. Verder heeft Leo voor het Exin opgaven ontworpen voor zowel het TMap NEXT Test Engineer als het Test Management-examen. Tevens is Leo lector van het lectoraat Software Quality & Testing aan de Fontys Hogescholen, lid van de normcommissie NC 381007 'Software and System Engineering' en Development Lead van de ISTQB Agile Add-On Syllabus.

Tenslotte is Van der Aalst een veelgevraagd docent voor testopleidingen, spreekt hij regelmatig op nationale- en internationale conferenties en is hij auteur van diverse artikelen.