Whittaker en Kamkolkar over het nieuwe testen

Tester moet ontwikkelaar beter maken

Dré de Man

Als deel van Visual Studio Team System 2010 heeft Microsoft Visual Studio Lab Management aangekondigd. Het is bedoeld om de samenwerking tussen ontwikkelaars en testers te verbeteren en het testen te vergemakkelijken. In dit gesprek met James Whittaker en Neelish Kamkolkar komt de nieuwe software aan de orde, maar gaat het toch vooral over testen en bugs in het algemeen.

James Whittaker hield zich zijn leven lang bezig met testen. Hij schreef zijn PhD al over model based testing, bracht het onderwerp als hoogleraar enorm onder de aandacht en werkt nu als software-architect binnen het Visual Studio team. Neelish Kamkolkar is senior product manager Vistual Studio Team Test.

Testen is onder ontwikkelaars niet populair. Het is net zoiets als naar de tandarts gaan: niet prettig maar als je het uitstelt heb je uiteindelijk een groter probleem. Een van de dingen die het onaangenaam maakt, is het zogenaamde bug ping pong: eindeloos heen en weer gaande communicatie over niet-reproduceerbare bugs.

Whittaker: "Meestal kan de tester het reproduceren op zijn machine, maar de ontwikkelaar niet op die van hem. Het is één van de belangrijkste oorzaken van ontwikkelingsvertragingen. Wat heel simpel zou moeten zijn, draait uit op uren en soms dagen werk en veel frustratie. Door de omgeving te virtualiseren kun je de garantie geven dat bugs te reproduceren zijn. Dat is het Team Labs-deel. Team Test heeft een rijke dataverzameling. We kunnen niet alleen de garantie geven dat de bug reproduceert, maar ook alle informatie erover aan de ontwikkelaar verstrekken, zodat hij veel sneller kan debuggen dan daarvoor."

Kamkolkar: "Een ander aspect ervan blijkt wanneer je naar het bouwen van applicaties kijkt vanuit het perspectief van lifecycle management. Iedere keer als er een verandering is, wordt er weer een build gemaakt en treden er heel veel wijzigingen op. Wanneer klanten gebruikmaken van continue integratie zijn er veel uitdagingen, omdat het moeilijk is dat proces snel af te ronden. We hebben het lab management geïntegreerd in het buildproces door de gelegenheid te bieden specifieke activiteiten in de build-definitie op te nemen. Daardoor kun je als de build af is, een virtuele omgeving ter beschikking stellen, de build op die virtuele machines deployen en er een snapshot van maken. Zodoende heb je altijd de beginsituatie waar je naar terug kunt gaan. Uiteindelijk kun je dan een set geautomatiseerde tests doen die het mogelijk maken er een betere kwaliteit build uit te halen, gezien vanuit de developer. Dit zorgt ervoor dat teams veranderingen beter kunnen omarmen omdat het build-proces efficiënter verloopt."

Helpt de grote populariteit van agile bij het populairder maken van testen?
Whittaker: "Wat er mijns inziens gebeurt, is dat testen over de gehele lifecycle gedistribueerd wordt. Ontwikkelaars voeren meer en meer verschillende soorten tests uit. Ze doen testdriven development, ze schrijven betere unittests en dat maakt testers vrij om zich te concentreren op het gedeelte van de applicatie waar de businesslogica zit, waar de echt belangrijke bugs zijn. De bugs waarvan we weten dat je die heel moeilijk in geautomatiseerde testen kunt vinden. We richten ons voor een belangrijk deel op het handmatig exploratief testen. Daar vind je de hoge kwaliteits-bugs, de bugs waarvan je niet wil dat ze in productie gaan."

En die heel duur zijn om op te lossen.
Whittaker: "Ja, bugs gerelateerd aan businesslogica vormen een groot probleem. Er waren ook geen tools die de testers hielpen ze te vinden. Ze worden niet door testscripts gevonden, maar door een mens die nadenkt over de businesslogica, gebruikmakend van zijn hersenen en zijn vingers. Die testers hebben tot nu toe in de kou gestaan, die hebben niet veel geautomatiseerde ondersteuning gekregen, weinig analytische tools, weinig hulp om hun problemen aan ontwikkelaars over te brengen. Exact de pijnpunten waar Team Test zich op richt."

Hoe doet u dat precies?
Whittaker: "Beter testcase-management is er een belangrijk onderdeel van. Het opnemen van een testcase en inchecken van

een test in het testcase-management-systeem. Zo verander je handmatige testcases in automatische testcases. En krijgen ze een eersteklas behandeling in de bug database, in de regression suites en in alles wat testers normaal geautomatiseerd doen. We nemen dus het verschil weg tussen handmatige en geautomatiseerde testcases. Het gaat om de vraag hoe een testcase past in het grote geheel. Als een requirement verandert of de code verandert, moeten testers hun testcases doorlopen en nadenken over de vraag hoe ze die code- of requirements-veranderingen testen. Door Team Foundation Server zijn requirements, code en testcases met elkaar verbonden. Als er nu iets verandert, weten testers waarop het betrekking heeft. Ze kunnen ze inchecken in hun regression suite en ze automatisch toepassen. Het begint dus als een handmatige tescase, het verandert in een regression testcase, maar niet alleen dat. Het wordt een geautomatiseerde testcase, gericht op deze specifieke verandering van de code of de requirements. Heel krachtig. We zien veel opwinding bij testers, omdat we ze behandelen zoals nooit tevoren."

Kamkolkar: "Om van de intensievere developer/testers-interactie te profiteren, is het echt nodig beter naar de testers te luisteren. Een van de manieren om big pingpong te vermijden bijvoorbeeld, is door de testers te helpen bugs beter op te slaan. Daarbij horen antwoorden op vragen van ontwikkelaars als: 'wat was je aan het doen?'. Als ik in de kamer naast je zou zitten, is het misschien gemakkelijk, maar wat als het andere team in een heel ander land zit? Als onderdeel van het opslaan van de bug vindt er ook meteen opslag plaats van alle teststappen, de testcase, de status van alle doorlopen stappen en een tijd-geïndexeerde video. Deze laat de developer precies zien wat hij deed. Samen met de datacollection krijg je ook de systeeminformatie en heb je de mogelijkheid checkpoints van de virtuele testomgeving te maken en die als een link op te nemen. Testers hoeven dus minder tijd te besteden aan het documenteren van de bugs en kunnen zich meer richten op het testen en het gevecht tegen bugs. Aan de andere kant krijgen de ontwikkelaars niet alleen met de bug de links naar de video, maar ook links om de virtuele omgeving te restoren naar het check point."

Whittaker: "We horen al van onze klanten dat ontwikkelaars minder bezig zijn met het vechten tegen bugs een meer met het schrijven van code. Dat testers meer tijd besteden aan testen dan

> 'We nemen het verschil weg tussen handmatige en geautomatiseerde testcase'

aan het stoeien met oude bugs. Ze doen dus meer productief werk en daar zijn we echt trots op. Want we hebben die productiviteit nodig, sofware is gecompliceerd. Hoe meer tijd ze echt bezig zijn met het schrijven van software in plaats van met het rommelen met al de kleine details, des te gelukkiger is iedereen."

Het testen van businesslogica kan heel gecompliceerd zijn. Een deel van de benodigde kennis voor het testen ervan zit bij ontwikkelaars, een ander deel bij de business. Voor een tester kan het heel moeilijk zijn om uit te vinden wat hij eigenlijk moet testen, wat de software eigenlijk zou moeten doen.
Whittaker: "We zijn de testers daarin op twee manieren tegemoet gekomen. Je kunt de testcases van tevoren schrijven. Maar je kunt ook de applicatie exploreren en een deel van de rijke mogelijkheden gebruiken om data op te vragen en op te slaan en zo de testcases documenteren. De testers zijn veel vrijer om de applicatie te exploreren en om te begrijpen hoe die werkt. Ze kunnen de applicatie beter belasten en gebruikmaken van de tooling-infrastructuur om de gecompliceerde testcases voor je te generen. Het tweede aspect is dat we het testen echt simpeler hebben gemaakt. Veel van de heel moeilijke technische aspecten ervan zijn niet meer noodzakelijk. Datacollection-mechanismen, recording-mechanismen, bugreporting, de tools doen dat voor je. Wij voorzien dat de businessanalisten - die nooit eerder getest hebben, omdat dat te moeilijk was - nu in staat zijn bij te springen. Zij gaan deel uitmaken van het software-testproces. We horen dat ook al van onze klanten. We zien een heel interessante interactie tussen businessanalisten en testers. De businessanalisten zijn er meer bij betrokken en daardoor kunnen de testers hun werk beter doen. Een van de doeleinden van Team System als geheel is ervoor te zorgen dat de mensen die de businesslogica begrijpen, beter betrokken zijn bij het proces van softwareontwikkeling. Als ze weinig anders doen dan ons aan het begin recuirements te geven, die we vervolgens links laten liggen omdat ze 'niet slim genoeg zijn om ons te helpen', doen we er uiteindelijk een gooi naar. Wij zijn in feite niet slim genoeg om het zonder hen te doen. Het betrekken van de businessanalisten bij de requirements, bij de ontwikkeling, bij het ontwerp, bij het testen tot aan het proces is één van de wezenlijke sterke punten van Team System als geheel."

Daar wil je uiteindelijk ook naar toe, naar het zoveel mogelijk betrekken van de businessanalisten en gebruikers bij het bouwen van de software.
Whittaker: "Juist. We hebben heel veel voordeel van het betrekken van businessanalisten en de eindgebruikers in het proces. De tools zijn ook eindelijk goed genoeg om dat deel van het proces, dat de eindgebruikers en businessanalisten haten, over te nemen."

Daardoor gaan ontwikkelaars echter nog steeds niet meer van testen houden.
Whittaker: "Developers mogen dan niet van testen houden, testers wel. Door deze mensen productiever te maken en beter te betrekken bij het ontwikkelproces, verklein je de last van het testen voor software-developers. Ontwikkelaars houden van de low level automation en coding-aspecten van het testen. We geven ze nu de mogelijkheid om dit te doen door de handmatige testers veel effectiever te laten zijn bij het testen van businesslogica. Er zijn twee soorten code in ieder ontwikkelproject. Businesslogica-code en de rest: infrastructuur, besturingssysteemspecifiek, security, performance, al die dingen. Test-automation en unit-testen

‘In plaats van het meten van de kwaliteit van de ontwikkelaars meet ik liever de testers in hun vermogen de ontwikkelaars beter te maken’

zijn heel goed in het testen van infrastructuurcode. Ontwikkelaars houden niet zo van – en zijn niet zo goed in – het testen van businesslogica-code. We maken het de softwaretesters nu mogelijk dat te doen. We doen ons best om de *sweet spot* te raken, we willen dat softwareontwikkeling niet alleen effectief maar ook aangenaam is. De testers moeten dus doen waar zij goed in zijn en wat hen plezier oplevert, en de ontwikkelaars net zo goed. We willen de vervelende dingen in de gehele levenscyclus wegnemen. Daar gaat het bij Team System om. We richten ons op de teams, op de rollen en op de integratie van die rollen van begin tot eind.”

Een ander aspect is de methodologie: in een traditioneel watervalproject wordt er helemaal aan het eind getest, dus in heel veel projecten is er helemaal geen tijd meer voor het testen.
Whittaker: “De testers zitten aan het einde van de waterval en uiteindelijk verdrinken ze (gelach). We moeten ze meer naar boven plaatsen, zodat ze vooral bezig zijn met zwemmen en minder met verdrinken.”

Maar als je in een traditioneel watervalproject werkt, zal testen altijd heel moeilijk blijven.
Whittaker: “Dat kan niet anders. Alle rotzooi komt aan het eind van het traject terecht, om het niet nog plastischer te zeggen. Als je eenmaal in de situatie zit dat je te laat bent en dat het budget op is, wordt testen geschrapt. Dat is niet acceptabel. Zorg er dus voor dat testers er eerder bij betrokken worden, zorg ervoor dat ze productiever zijn, automatiseer de saaie gedeelten, en help bij het creatieve deel. Dat is het idee.”

Werkt u zelf agile?
Whittaker: “Ja wij zijn een agile-team. We testen agile en we proberen veel dingen tijdens het ontwikkelen die in de tools terechtkomen. We gebruiken onze eigen tools om deze onderdelen te testen. Zo komen we met betere manieren om te werken en brengen die kennis in de tools.”

‘Het is verleidelijk om te zeggen welke ontwikkelaars de meeste bugs schrijven, maar dat is niet echt productief’

Heeft uw team net als dat van Gerd Drapers ook een bug jail, een gevangenis voor ontwikkelaars die te veel bugs schrijven, waar ze in moeten blijven totdat ze hun eigen bugs opgelost hebben?

Whittaker: “Ja, dat is geen plaats waar je graag terechtkomt.”

Maar zo’n feature zit nog niet in het nieuwe tool?
Whittaker: “Nee, er zijn altijd onderdelen die wij al gebruiken maar die niet in de tools zitten. Pas na verloop van tijd is de nieuwe versie goed genoeg om zichzelf te schrijven, dus we lopen steeds een beetje voor. Bug Jail is een goed voorbeeld van de zaken die ons ertoe aanzetten te doen wat we nu doen. Als een ontwikkelaar een bug niet kan reproduceren, deleten we hem niet, we weten dat het er eentje is. Dat maakt deel uit van het in de bug jail zitten, je hebt een aantal niet reproduceerbare bugs en de ontwikkelaars werken zich gek om ze op te lossen. In het ideale geval zou geen enkele ontwikkelaar ooit in de bug jail belanden. In die situatie zou de snelheid waarmee de code door het systeem gaat enorm verbeteren. Eén van de dingen die we bijhouden terwijl we onze tools verbeteren, is of we een reductie zien van het aantal VS-ontwikkelaars dat in de bug jail terechtkomt. Als dit niet gebeurt, moeten we onszelf een paar pijnlijke vragen stellen. Maar we zien die reductie wél en analyseren de uitkomsten iedere dag. De antwoorden vloeien terug in onze tools. Als iemand naar de bug jail gaat, wil je dat iemand daar zit om redenen die goed zijn voor de productiviteit. We noemen het niet bug time-out, dat zou te *weeny* zijn, iets voor watjes. Niemand zou ons serieus nemen, het moet bug jail zijn. Time-out is voor vijf jaar oude kinderen, dit is echt veel ernstiger.”

Laten we eens kijken naar de persoonlijke aspecten van bugs. Als je informatie verzamelt over bugs, krijg je ook informatie over de ontwikkelaars. Niet iedereen is even goed bijvoorbeeld.
Whittaker: “Geen twijfel over mogelijk, sommige ontwikkelaars zijn beter dan andere en sommige testers zijn ook beter dan hun collega’s. We investeren niet een heleboel tijd in het bestuderen van de data om te zeggen: één specifieke ontwikkelaar is bijzonder slecht. Ons doel is ze beter te maken. Als we weten dat er een patroon zit in het soort bugs dat ze schrijven, dan gaan we praten. Binnen de testgemeenschap hebben we een discussie over de vraag wat iemand tot een goede tester maakt. Als je meer bugs vindt dan iemand anders? Ik zou zeggen, nee, je bent een betere tester wanneer de mensen van wie je code test, sneller beter worden dan andere ontwikkelaars. Zo meet ik mijn testers graag. In onze groep is het zaak voor de testers om patronen bij de ontwikkelaars te zien. Hier heb je een ontwikkelaar die iedere keer weer opnieuw dezelfde bug schrijft. Dat is het moment waarop in mijn team de tester de verantwoordelijkheid heeft naar de ontwikkelaar toe te stappen en te zeggen: ‘Dit zijn de testcases die ik op jouw code loslaat, en jij loopt hierdoor het gevaar in de bug jail te belanden’. De ontwikkelaar neemt meestal die uitdaging aan en zorgt ervoor dat die testcases geen bugs meer vinden en daar willen we precies naartoe. In plaats van het meten van de kwaliteit van de ontwikkelaars meet ik liever de testers in hun vermogen de ontwikkelaars beter te maken. Ontwikkelaars zouden iedere testcyclus beter moeten worden.”

In het ideale geval zou je je kunnen voorstellen dat VS Team system bij een bepaalde ontwikkelaar zou suggereren dat hij een training moet volgen, of iets anders dat de kwaliteit van zijn code zou verbeteren.
Whittaker: “Hé, niet onze features voor de volgende versie weggeven! Maar serieus, als de software zo goed zou zijn, zou je nog een stap verder kunnen gaan en de bug herstellen, *self healing* software.”

Denkt u wel eens aan dat soort zaken?
Whittaker: “Jazeker. Microsoft is erg rijk aan data en daar zijn veel patronen in te ontdekken. Wij werken samen met een aantal onderzoekers van Microsoft Research. Kijk, het is verleidelijk om

te zeggen welke ontwikkelaars de meeste bugs schrijven, maar dat is niet echt productief. Welke bugs treden op die je met wat eenvoudige training echt zou kunnen oplossen, is een betere vraag. Of met andere technieken of misschien zelfs met nieuwe regels in Visual Studio die voorkomen dat die bugs überhaupt geschreven worden. Al die analyse vindt voortdurend plaats. Wij hebben researchers die de patronen in de bugs identificeren maar wij schrijven de tools om ze te voorkomen. In 2001 en 2002 gebeurde het ook met security. Het patroon was: wij schrijven security bugs. Steeds opnieuw. Niet met opzet, ontwikkelaars realiseerden zich niet wat ze deden. Het ging om een klasse bugs waar we daarvoor nooit over hadden nagedacht. We hebben dat opgelost. Nu proberen we dit toe te passen op iedere belangrijke klasse bugs die we tegenkomen. Vanuit het perspectief van Visual Studio is dat onze job. Wij schrijven de tools die anderen gebruiken om software te schrijven. Als we betere sofware willen, moeten onze tools in dat proces hulp bieden. Als wij hulp bieden om te voorkomen dat bugs geschreven worden, is dat veel beter dan tools te bouwen die helpen bugs te vinden."

De meeste lezers van Microsoft .Net Magazine zijn ontwikkelaars, geen testers. Zou u ze een advies kunnen geven, een top tien om bugs te vermijden?
Whittaker: "Ik weet niet of ik tot tien kom zo uit mijn hoofd. De eerste: schrijf niets waarvan je niet weet hoe het valt te testen. Ik vind dat erg belangrijk. Denk voortdurend aan testen tijdens het ontwikkelproces. Een andere is: leer altijd van je fouten. Laat een bug nooit ondergewaardeerd blijven, leer ervan zodat je ze niet opnieuw schrijft. Ieder bug moet een leerervaring zijn. Werk met je tester samen. Vergeet nooit je gebruiker. Daar is een geweldig citaat over. Stel je bij het schrijven van ieder stukje code voor dat de vent die de code onderhoudt een psychopaat is die voortdurend met zijn geweer loopt te zwaaien. En die weet waar je woont."

Interesse?

Op 8 januari, 19 februari en 16 maart 2009 organiseert Microsoft een VSTS 2010 Intrack. Zie pagina 46 of http://www.msdnintracks.com/msdnintracks voor meer info.

Tekst en foto's: **Dré de Man**