VSTS werkt lekker, maar is nog niet af

Meer aandacht voor testen en werkoverdracht

Gerard van der Pol en Robert de Ruiter

Sam Guckenheimer, Group Product Planner Visual Studio Team System bij Microsoft Corporation, was in Nederland voor het Application Platform Optimization Congres. Een mooie gelegenheid voor .NET Magazine om hem aan de tand te voelen over de jongste ontwikkelingen rond VSTS. Wat staat er aan te komen in 2010? En wat kunnen we daarna nog verwachten? Guckenheimer vertelt.

Een kleine, vriendelijke man. Keurig getrimde baard onder een al even keurig getrimd kalend hoofd. De donkere ogen kijken je vanachter de brillenglazen observerend aan. Hij vertoont geen spoor van vermoeidheid ondanks het feit dat hij een dag erder pas op Schiphol is aangekomen en sindsdien al drie keer als spreker op het congres is opgetreden. Zo'n interview kan er best nog bij, meldt hij gemeend enthousiast. En ach, het werk zit er bijna op. Vanavond nog een goede 'company'-maaltijd met een aantal genodigden en dan nog twee dagen de toerist uithangen voor hij weer in het vliegtuig stapt. Maar eerst leggen wij nog een uurtje beslag op hem. Hoe zit dat nu met VSTS? Wat kunnen we volgend jaar verwachten?

SAM GUCKENHEIMER.

"We streven naar verdere optimalisatie ten aanzien van het testen van software. In 2010 introduceren we Test Impact Analysis. Deze feature heeft twee gezichten, namelijk één voor de ontwikkelaar en één voor de tester. In Visual Studio adviseert het de ontwikkelaar welke set van unit tests uit te voeren op basis van de veranderingen in de code. Dezelfde technologie wordt ook ingezet ter ondersteuning van de activiteiten van de tester in wat nu nog Camano heet. De feature heet Pick a Build," steekt Guckenheimer van wal.

Op basis van het verschil tussen de build die de tester nu gebruikt en diegene die hij wil gaan gebruiken wordt er ook een advies gegeven. Dit advies kan ofwel requirements gericht zijn, dus een advies welke user stories te testen of gericht op de test cases die geraakt zijn. Het geeft ook een antwoord op de vraag of het zinvol is om al een nieuwe test cyclus te starten. "We weten dat er een nieuwe build beschikbaar is, maar heeft deze nieuwe build de verwachte nieuwe functionaliteit die we willen gaan testen." Het gaat er uiteindelijk om je tijd te investeren in activiteiten die het meeste waarde toevoegen.

Na 2010

Guckenheimer schetst ook een aantal zaken, waar aan wordt gedacht voor de release na 2010. "Gedurende het ontwerpproces heb je voortdurend te maken met wijzigingen in de teamsamenstelling. Dat komt in deze tijd met in- en outsourcing, acquisities en zo meer steeds vaker voor. Goed en snel het werk kunnen overdragen, is dus erg belangrijk.




Lean en agile

Als gevolg van de economische situatie letten klanten meer op de kosten en snelle resultaten. Kosten kunnen we terugdringen door overbodige handelingen en code uit het ontwikkelproces te halen. Dit levert een enorme tijdsbesparing op. Je moet je concentreren op die zaken, die de klant waardevol vindt. Toyota werd met haar 'lean manufacuring' systematiek niet alleen succesvol door de 'waste' uit het productieproces te filteren, maar ook door die auto's te bouwen, die de klanten wilden kopen. Je moet je dus concentreren op lean software-engineering en een lean application lifecycle. Daar gaat Team System over. We helpen onze klanten om de flow of value voor hun klanten te verbeteren.

Het idee om klant en ontwerper dichter bij elkaar te brengen is al zo'n acht jaar oud, toen het begrip agile werd gelanceerd. Deze methodiek gaat er onder meer van uit dat het ontwerpteam samen met de klant aan de slag gaat. Maar dat is voor Microsoft met zijn miljoenen klanten niet realistisch. Het is in het algemeen niet realistisch om te veel op de klant te leunen, vindt Guckenheimer, omdat deze zijn eigen leven en zijn eigen werk heeft. "De klant is geen software ontwikkelaar. De interactie met de klant moet dus gaan over iets wat deze begrijpt. Tegenwoordig gaan we ervan uit dat we een klein deel van de programmatuur neerzetten en dan aan de klant vragen 'is dit wat je wilt hebben'. Vanaf dat punt bouwen we verder. Zo ga je van batch naar batch. Je corrigeert tijdens de ontwikkeling en voert wijzigingen door waar nodig".

We proberen voor de ontwikkelaar patronen voor het gebruik van de code in beeld te brengen. Ik zie een stuk code en ik zie een methode. Als ik hier iets mee doe, wat wordt er dan verder beïnvloed. Het kost nu veel tijd om je in deze materie in te werken. We hebben geleerd dat we de historische TFS data kunnen gebruiken om dit zoekwerk te verlichten. Je krijgt bovendien veel grotere mogelijkheden tot hergebruik. Het probleem van hergebruik is dat degene die de investering doet in het schrijven van herbruikbare code, daar zelf vaak weinig voordeel uit haalt. De motivatie om herbruikbare code te schrijven is hierdoor laag.

Een andere uitdaging is bepaalde patronen in code te herkennen en dit, indien nodig, dan te gaan refactoren. Code Analysis en Code Metrics leveren hier een belangrijke bijdrage. In principe zouden deze processen op de achtergrond continu actief moeten zijn en meer een adviserende rol dan een controlerende rol moeten hebben. Probleem is dat beide activiteiten computer intensief zijn. Vergelijkbaar is de spellingchecker in Word, het is heel nuttig, maar de feature zou minder bruikbaar zijn indien je na elk woord 20 seconden zou moeten wachten, totdat de spelling is gecontroleerd. Een oplossing voor dit probleem is wellicht het gebruik van Cloud computing.

> De Office Divisie vertelde aan iedereen hoeveel prettiger hun werk was geworden.

Een ander voorbeeld is het uitvoeren van code reviews. Vele organisaties zien het voordeel hiervan, maar hebben gewoonweg niet de tijd en resources beschikbaar om dit goed in te zetten. Wat als we deze activiteit zouden kunnen ondersteunen middels de kennis die we al hebben van de code, namelijk de historische data in TFS, en daardoor de benodigde effort kunnen reduceren.

De nieuwe release na 2010 zal daarnaast weer de nodige nieuwe items laten zien gebaseerd op het werk van Microsoft Research," aldus Guckenheimer.

Anders werken

Tot dusver is hij tevreden over de resultaten van VSTS. In de eigen geledingen is het niet zonder slag of stoot ingevoerd, maar wel geaccepteerd en wordt het steeds enthousiaster toegepast. "Onze eigen ontwikkelaars kunnen het best weergeven of we met Team System op de goede weg zijn. De Office Divisie was in het begin heel sceptisch. Er werd van bovenaf opgelegd dat men met VSTS moest gaan werken. We hebben toen gezegd: als je het hebt gebruikt en een aantal stappen van het programma hebt doorlopen en je dan nog kritiek hebt, geloven we je. Na ongeveer drie maanden ontstond een patroon waarbij wij niet meer spraken, maar de gebruikers van de Office Divisie. Zij vertelden aan iedereen hoeveel prettiger hun werk was geworden en hebben zelf nog een aantal nieuwe ideeën ingebracht. Door hun kennis van Excel, Excel Services en SharePoint hebben we het dashboard kunnen verbeteren."

"Het is altijd moeilijk – en met name bij Microsoft – om mensen te vragen anders te gaan werken. Microsoft neemt mensen in dienst die vol zelfvertrouwen zijn en hun eigen ideeën over development hebben. Die zelf beslissingen willen nemen. Zij zijn dus heel sceptisch als je hen vertelt dat ze dingen beter kunnen doen. We hebben nu eenmaal geen top-down structuur."

"De bottlenecks zaten aan twee zijden van

de ontwikkelaar. Voor we VSTS gebruikten, kwamen de releases als pindakaas uit de development-molen. Niemand wist welke features waren gebouwd, hoe ze werkten of waar ze goed voor waren. In VSTS zit een planningproces waardoor iedereen weet wat we willen gaan bouwen. Dit wordt uitgedrukt in customer-value en leidt tot een aantal deliverables. Het ontwikkelteam heeft een grote mate van vrijheid om dit te bouwen, maar ze weten dat ze met zijn allen verantwoordelijk zijn voor de kwaliteit van het product en dat die deliverables op tijd moeten worden opgeleverd."

"Bij de ontwikkeling van VSTS 2008 hebben we niet vooraf alle requirements gespecificeerd, maar we hebben een algemeen ontwerp neergelegd en de verfijning door de feature crew laten doen. Het algemene ontwerp is vooraf besproken met key stakeholders. Dat levert een basis voor het programma op, waarop je verder kunt werken. Als de basis er ligt kijk je wat je nog mist en ga je dat bouwen".

In eerste instantie is VSTS niet gebouwd voor eigen gebruik. In 2003, toen Guckenheimer bij Microsoft in dienst trad, lag er het idee om een nieuw product voor de klanten op de markt te brengen.

Silo's

In de jaren tachtig en negentig waren mensen voornamelijk gericht op hun eigen werk. Ieder werkte in zijn of haar eigen silo. Veel werk werd weer weggegooid, omdat de stroom tussen de silo's niet goed was georganiseerd. "We wilden veel al aanwezige tools gebruiken om de verschillende silo's met werk bij elkaar te brengen. Daarom is alle aandacht gericht op het terugbrengen van de verliezen die ontstonden, omdat de communicatie tussen de silo's niet goed verliep. We hebben ons dus gefocust op verbetering van de stroom en de waste uit het ontwikkelproces te halen.

We hadden het voordeel dat we binnen Microsoft veel assets hadden die we konden gebruiken. De technologieën op het gebied van codeanalyse en coverage en performance profiling waren direct gebaseerd op bestaande technologieën van Microsoft Research. De core functionaliteit van VSTS hebben we vanaf de grond opgebouwd. Een ander voorbeeld van het hergebruiken van technologie, is wat we hebben gedaan voor Lab Management waarin een build autmation extensie zit om een virtueel lab te produceren. Een testomgeving, die werkt met virtual machines. Er zijn dus veel gevallen waar we Microsoft technology assets hebben gebruikt."

Binnen Microsoft bestond de behoefte om de development te stroomlijnen. VSTS moest de divisies met al hun individuele ontwikkelaars een tool in handen geven om het werk plezieriger en eenvoudiger te maken. De vele tools die beschikbaar waren en elkaar overlapten moesten in één overzichtelijk tool worden ondergebracht. Bovendien lag er de behoefte om een sterkere band tussen Microsoft Research en de ontwikkelafdelingen te creëren. Het primaire doel met VSTS was om een product voor derden neer te zetten, maar hier begon ook de dogfooding.

Dogfooding

Dogfooding is in meerdere opzichten een mooi ding, vindt Sam Guckenheimer. In tegenstelling tot enkele andere productcategorieën is Microsoft hier ook gebruiker van software. "We maken software en daar hebben we software voor nodig. In TFS zit een gemeenschappelijke check-in dialog waar je kunt zien welke files veranderd zijn. Uit onze observaties bleek dat er in 2003 nog zeventien handelingen nodig waren voor het inchecken van files en afhandelen van de workitems die verder het proces in kunnen. Dat hebben we teruggebracht tot twee. Ook de shelveset is ontstaan uit interne observaties. Het probleem met branching en sourcecontrol systemen was dat de developers veel tijdelijke branches aanmaakten, die het systeem gingen verstoppen. Daar zaten ook simpele activiteiten bij, die helemaal geen branches nodig hadden. Nu kun je het werk aan een bepaald project dat je moet onderbreken op de server opslaan in een shelveset. Dat houdt de voortgang van het project niet tegen.

De grootste verandering in 2010 wordt 'No more no repro'.

Een ander voorbeeld is Codecoverage highlighting in de editor. We hadden verschillende code coverage tools, die als batchtools werkten met veel output als resultaat. Als developer moest je al die resultaten bestuderen om de codecoverage te beoordelen en om te weten wat je moest doen. We hebben toen besloten om deze informatie als het ware in de editor middels kleuren te tekenen. Zodat je het ziet. Je kunt met een rechter muisklik direct actie ondernemen. Dat zijn kleine dingen die veel overbodige handelingen wegnemen."

'No more no repro'

De grootste verandering in 2010 wordt 'No more no repro'. "We hebben gezien dat een van de grootste veroorzakers van waste tegenwoordig de manier is waarop met bugs wordt omgegaan. Iemand rapporteert een bug, die vaak niet kan worden gereproduceerd. De developer reageert dan met:

'Ik weet niet wat je rapporteert. Misschien geloof ik je niet en ben je te dom om de software te begrijpen. Misschien ook rapporteer je iets anders dan er werkelijk gebeurt. Misschien geloof ik je wel, maar ik weet niet hoe ik een experiment moet opzetten met de juiste initiële condities, die voor mij kunnen reproduceren wat jij zag.' Dit komt bijvoorbeeld voor als het om een concurrency-probleem gaat, waarbij twee threads elkaar dwars zitten."

We kunnen meer waarde in het product stoppen omdat we minder werk hebben aan bugs.

We voorzien in zes mechanismes in VSTS 2010 om het repro-probleem te tackelen:

- Er wordt een action-log aangemaakt, dat precies de handelingen van de ontwikkelaar vastlegt;
- Van het action-log wordt een geindexeerde video gemaakt. Je ziet dat op 0 minuten en 40 seconden dit venster in de browser verschijnt en je constateert dat het venster niet weergeeft wat je had verwacht. Dit rekent af met twijfel over de gerapporteerde bugs;
- Er worden snapshots gemaakt die je kunt annoteren. Dat kan iets duidelijks zijn zoals 'dit is de verkeerde prijs', maar ook iets subtiels zoals 'je gebruikt het verkeerde logo'. De ontwikkelaar weet wellicht niet welk logo hij moet gebruiken;
- De eigenschappen van de machine waarop de software draait worden vastgesteld;
- We capturen een trace. De ontwikkelaar klikt op de bug, dubbelklikt op de trace en komt dan in een debugging sessie terecht. Nu is het nog zo dat de debuggingsessie moet beginnen door de software te starten en te proberen het experiment te reproduceren. Dit is een debuggingsessie op basis van de data toen de fout werd gerapporteerd. Het is dus een soort digitale video-opname die precies vastlegt wat er gebeurt, zoals de black box in een vliegtuig. Zo zie je in de debugger dat net voor de 'pagina niet gevonden' optreedt er een ongeldige query op de database wordt losgelaten. Die query is ergens opgevoerd. Je kunt nu terug naar dat punt en daar de variabelen bekijken die de bug veroorzaken. Deze vorm van offline-dubugging achteraf is prachtig, want je hoeft als ontwikkelaar niet steeds weer experimenten op te zetten;
- Als de tests worden gerund kan de tester met gebruik van het virtual lab de staat van de machines bestuderen op het moment, waarop de bug optreedt. Dat maakt het mogelijk om te bekijken of het werkelijk de query was, die de bug veroorzaakte of dat er op het moment waarop de bug optreedt de database een verkeerde referentie teruggeeft.

Al deze mechanismes samen hebben de potentie om de waste uit te bannen. Je hoeft ook niet meer alle developers bij elkaar in een ruimte te zetten. Het maakt niet uit of ze naast elkaar of driehonderd kilometer uit elkaar zitten. Of misschien wel in India of China. Verschillen in taal of cultuur doen er niet meer toe. Als je via email communiceert, weet je dat er miscommunicaties ontstaan. VSTS ondervangt dat allemaal.

"De verwachting is dat we hierdoor heel veel tijd besparen voor de ontwikkelaar en tester. Misschien wel zoveel als 25 procent. Ik heb het dan niet over het aantal gevallen dat wordt gerapporteerd, maar om de activiteiten die veel tijd kosten. In ons geval zal dit eerder omgezet worden in meer ontwikkeling dan in een kortere time-to-delivery. We zullen in staat zijn om meer waarde in het product te stoppen omdat we minder tijd hoeven te besteden aan bugbestrijding. Dit verschilt natuurlijk van case tot case."

Multicultureel Amerika

De ouders van Sam Guckenheimer zijn in 1937 vanuit het toen al roerige Duitsland naar de Verenigde Staten gevlucht. Zelf is hij een rasechte Amerikaan met als enige 'afwijking' dat hij de Duitse taal beheerst. Hij is getrouwd met Monica Kleinjans. De (achter)grootouders van de echtgenote van Sam hadden hun wortels in Nederland. Ze vestigden zich in Holland in Michigan, waar ze tot op de dag van vandaag nog een tulpenfestival hebben. Je vindt er ook windmolens, klompen en Delfts blauwe tegeltjes op de muren. Veel kerken zijn er Nederlands Hervormd. "En ze eten aardappelsla", suggereert Sam met een brede grijns en een perfecte uitspraak van 'aardappelsla'.

Sam en Monica hebben drie kinderen, waarvan er twee zijn geadopteerd. Deze kinderen komen uit Cambodja. Toen ze werden geadopteerd woonden de Guckenheimers in Massachusetts, waar iedereen er direct van uitging dat de kinderen geadopteerd waren. Toen Sam bij Microsoft ging werken verhuisde het gezin naar de omgeving van Seattle aan de westkust van de VS. Daar zijn sterke Aziatische invloeden. De mensen daar gingen er gewoon vanuit dat de geadopteerde kinderen zijn eigen kinderen waren. "Great. We found the right place!" zegt hij daar nu over. "That is America".

Gerard van der Pol, is VSTS technical specialist bij Microsoft.

Robert de Ruiter, is hoofdredacteur van .NET Magazine