Testen in de Application Lifecycle

Visual Studio 2010 Test Edition

Rob Kuijt en Clemens Reijnen

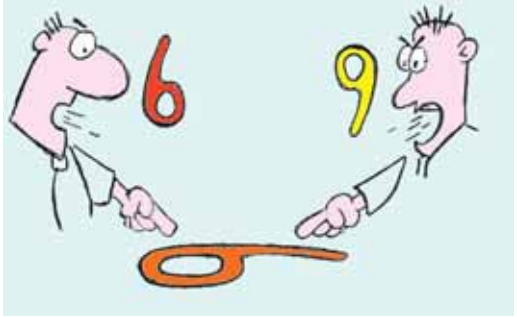
Visual Studio Test Edition wordt in de komende versie [2010] significant uitgebreid. Naast de ondersteuning voor de ontwikkelaar- en systeemtester biedt deze versie tevens ondersteuning voor de gebruiker/acceptatie testers. Hiermee dekt de komende versie van Visual Studio het complete spectrum van testen.

Dat testen een belangrijk onderdeel is bij het ontwikkelen van software weet iedereen. Dat de testafdeling compleet afgescheiden is van de afdelingen die de software ontwikkelen is ook meer uitzondering dan regel. Deze scheiding tussen de twee werelden is niet alleen fysiek, maar is ook aanwezig op het gebied van tooling. Samenwerking is ver te zoeken, terwijl de samenwerking van ontwikkelaars en testers door een gestructureerde testaanpak tal van voordelen biedt. Het verhoogt de efficientie van de hele application lifecycle en de kwaliteit van het product wordt aanzienlijk verbeterd, wat een kostenreductie met zich meebrengt. In dit artikel krijgt u te zien hoe Visual Studio 2010 het softwareontwikkelproces en samenwerking tussen ontwikkelteam en testers aanzienlijk verbetert.

Een voorbeeld uit de praktijk.

Samenwerken gaat niet vanzelf. Van nature lijken ontwikkelaars en testers de neiging te hebben om elkaar niet te (willen) verstaan. Vanuit die houding is het lastig om erachter te komen dat je elkaar nodig hebt om goede kwaliteit software te leveren.

Het gevolg van deze virtuele (en soms echte) muur is desastreus voor de kwaliteit. Vooral door aannames en interpretaties van de, over het algemeen, niet overduidelijke specificaties ontstaan veel bugs. Overleg over deze bugs kost veel tijd, vooral als ze ook nog eens niet reproduceerbaar blijken te zijn.



... HEBBEN DE NEIGING OM ELKAAR NIET TE VERSTAAN

Na een ontwikkeltraject van vijf manjaar wordt een test uitgevoerd door een team van testers. Het testteam gaat voortvarend aan de slag met de testgevallen die ze tijdens het bouwen hebben voorbereid. Na drie maanden komen ze met hun oordeel: Negatief vrijgave advies, vanwege (8) blokkerende en (22) grote fouten adviseren ze het systeem NIET in productie te nemen.
Zoals te begrijpen is dit een stevige teleurstelling voor zowel het projectteam als de opdrachtgevers en dus wordt er een taskforce ingesteld om de ontstane vertraging en dus schade binnen de perken te houden. Na analyse blijkt het allemaal wel mee te vallen. Na filtering van testfouten (20%), wensen (20%) en wijzigingen (die niet aan de testers zijn doorgegeven; 30%) blijken de herstelkosten circa tweehonderd uur te zijn (de resterende 30%).
Al met al had het bouwteam het dus eigenlijk goed gedaan. Toch zou bij betere communicatie van beide kanten, voor en tijdens de testuitvoering, het eindoordeel waarschijnlijk positief en dus een FEEST zijn geweest. Heel wat anders dan de koude douche!

Samenwerking in de Application Lifecycle

Dit typische voorbeeld laat zien dat de samenwerking tussen testafdelingen en ontwikkelafdelingen vaak nog ver te zoeken is. Als er geen energie wordt gestoken in de samenwerking tussen ontwikkelaars en testers ontstaat vanzelf de spreekwoordelijke 'muur'. Over het algemeen zorgt deze muur voor onbegrip, irritatie en geforceerde afbakening van taken en dus tijd- en kwaliteitsverlies. Het zogenoemde 'ping pong' effect met (niet reproduceerbare) bevindingen is hiervan het meest aansprekende en voorkomende voorbeeld.

Softwareontwikkeling en alle rollen die hierbij betrokken zijn dienen één enkel doel na te streven: toegevoegde waarde leveren voor de klant. Het maakt niet uit wat voor soort applicatie er gerealiseerd dient te worden. Alle werkzaamheden en alle rollen dienen toe te werken naar dit ene doel. De samenwerking tussen rollen, processen en hulpmidden wordt ook wel samengevat in de naam Application Lifecycle Management [ALM]. Termen als accountability, governance en compliance worden gebruikt wan-

De hoge kwaliteitseisen en korte time-to-market voor de informatiesystemen vereist een draai van 180 graden met betrekking tot het testdenken

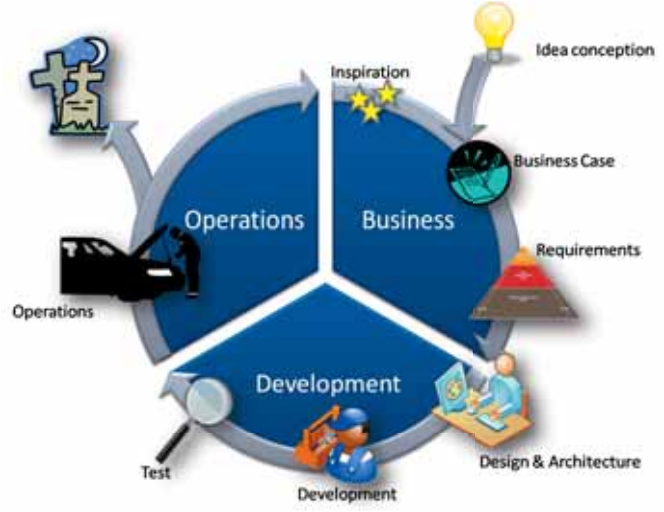
neer over ALM wordt gesproken, maar al deze termen refereren naar de simpele term samenwerken.

Wanneer ontwikkelaars en testers meer inzicht geven in elkaars werk en uitdagingen, ontstaat er meer onderling begrip. Irritaties verdwijnen en samen wordt er aan het gezamenlijke doel gewerkt. Visual Studio Team System 2010 gaat deze samenwerking beter ondersteunen. Door de tester hulpmiddelen te geven bij het uitvoeren van zijn/haar taken en first-class-citizen te maken binnen de Team Foundation Server. Waardoor hij/zij inzicht heeft en inzicht geeft in werkzaamheden die uitgevoerd worden, hoever deze taken zijn en welke stappen er nog genomen moeten worden. Naast dit inzicht in elkaars werk, wat vaak al een aardige cultuuromslag betekent, heeft de tester ook beter inzicht en toegang tot de ALM-Artifacts van de andere rollen. Bijvoorbeeld de use case en activity diagrammen van de business- en informatie analisten en de andere diagrammen van de designers en architecten zijn eenvoudig beschikbaar via de Visual Studio Team Architect Edition. Er wordt niet alleen samengewerkt door inzicht te geven in werkzaamheden, maar er wordt samengewerkt aan gezamenlijke producten.

Om duidelijk te krijgen hoe Visual Studio Team System 2010 de tester gaat ondersteunen en zorg draagt voor een betere samenwerking moet eerst gekeken worden naar de doelen en taken die testers nastreven en uitvoeren.

Testdoelen

In de tijd van de monolieten was het mogelijk om het testen uit te stellen tot de acceptatietest. Doordat in een monoliet alle functionaliteit compact in één groot component bij elkaar aanwezig was, waren fouten snel op te sporen en snel te herstellen. Die tijd is voorbij. Als je in de huidige architecturen teveel op de eindcontrole gaat leunen, is de kans aanzienlijk dat je de gewenste on-time, on-budget oplevering niet gaat halen.



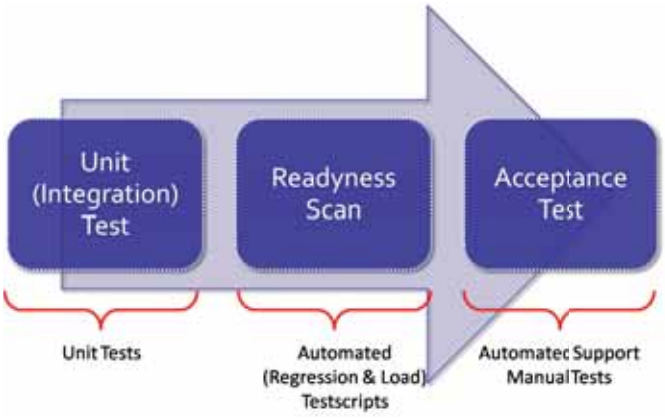
APPLICATION LIFECYCLE MANAGEMENT

De hoge kwaliteitseisen en korte time-to-market voor de informatiesystemen vereist een draai van 180 graden met betrekking tot het testdenken. In plaats van uitstellen moet nu de mindset op 'Finding Bugs as early as possible'.
Elke fase van het ontwikkelproces kent zijn eigen testdoelen en diepgang van testdekking. De diepgang is vooral afhankelijk van de risico's: des te hoger de kans op schade, des te beter dient de testdekking te zijn. De testdoelen zijn vooral gericht op het vinden van de fouten die in die fase ('as early as possible') gevonden kunnen worden:

- in de unit test wordt de interne logica van de unit getest,
- de unit integratie test dient aan te tonen dat de units elkaar 'verstaan',
- de systeemtest (readyness scan genoemd in de Acceptance Test Guidance van de Patterns and Practices Group, zie nuttige links) dient aan te tonen/te demonstreren dat het systeem voldoet aan de overeengekomen functionaliteit,
- afsluitend kijkt de acceptatietest naar de inpasbaarheid van het systeem in zijn uiteindelijke omgeving.

De testdoelen en (diepgang van) testdekking per fase worden op elkaar afgestemd middels een Master Test Plan (zie TMap® www.tmap.net).

Elke test moet kunnen vertrouwen op eerder uitgevoerde tests, alleen dan blijft de complexiteit van fouten beperkt tot de fouten die bij de laatst genomen stap zijn geïntroduceerd. Het instellen van een leerlus maakt het mogelijk om eventuele tekortkomingen in deze (test)procesketen te corrigeren (zie verderop in dit artikel). Ook de inzet van tooling speelt een cruciale rol. Handmatig uitvoeren van testgevallen, in de tegenwoordig veel toegepaste Agile werkwijze, is niet haalbaar. Dus automatische ondersteuning is een must.
De ervaring leert dat klakkeloze inzet van tools meer chaos dan winst oplevert. Vanaf het begin dienen de tests gestructureerd te worden opgezet om de inspanningen met betrekking tot het beheer binnen de perken te houden.



TESTTYPES BINNEN DE APPLICATION LIFECYCLE

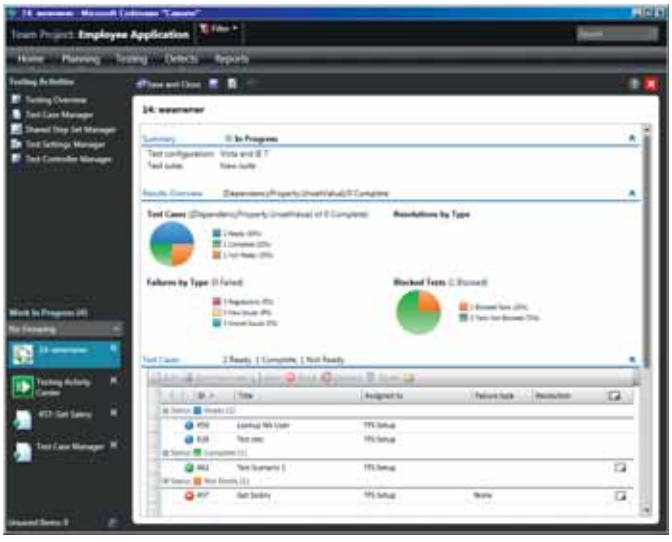
Ieder fase van het development proces en bijhorend testdoel kent ook een specifieke rol met specifieke vaardigheden en taken. Deze rollen worden ondersteund door specifieke onderdelen van VSTS 2010.

Samengevat zijn er de volgende typen testers:

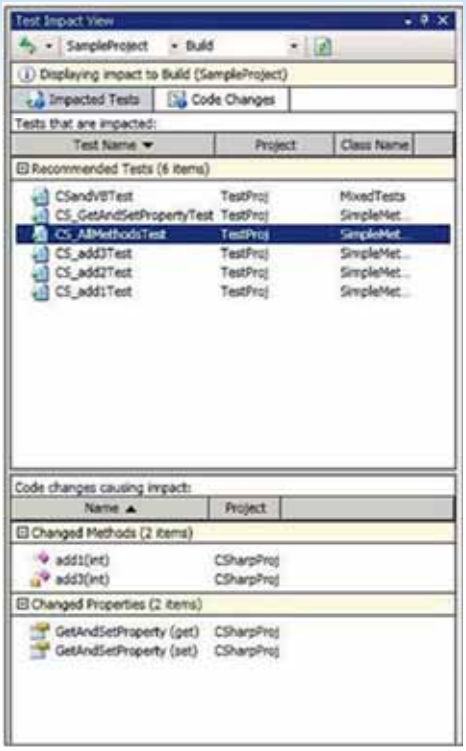
De ontwikkeltester
Deze kent de interne werking van het systeem en maakt gebruik van test-frameworks om de tests, die hij wil uitvoeren, te automatiseren. Deze testrol die vaak, zo niet altijd, uitgevoerd wordt door ontwikkelaars, wordt al geruime tijd ondersteund met technische hulpmiddelen als Unittest frameworks en de huidige versie van Visual Studio. Deze functionaliteit van VSTS zit niet alleen in de Test Edtion maar is ook volledig aanwezig in de Developer Edition, hierbij bevestigend dat deze taken hoofdzakelijk uitgevoerd worden door ontwikkelaars. VSTS 2010 maakt het leven van de ontwikkeltester eenvoudiger door intelligente ondersteuning van Test Driven Developement.

De systeemtester
De systeemtester test het systeem op volledigheid, of aan de eisen is voldaan. Deze eisen zijn niet alleen de functionele eisen maar ook kwaliteitseisen zoals performance en beveiliging. Systeemtesters maken veel gebruik van scripts om de werkzaamheden te automatiseren, hij/zij wil dezelfde tests kunnen uitvoeren op verschillende infrastructuren en configuraties. VSTS kent al geruime tijd de Load- en Webtesting functionaliteit binnen de Test Edition met de komst van VSTS 2010 wordt het maken van testscripts vereenvoudigd door de toevoeging van het zogenoemde 'CodedUI' framework.

De acceptatietester
De acceptatietester, ook wel gebruikertester of 'black box tester' genoemd, is ook daadwerkelijk een tester die gebruik van de applicatie test en niets weet van de interne werking van de applicatie. Hij/zij maakt testcases aan de hand van de functionele eisen, hierbij gebruikmakend van verschillende testtechnieken.
De acceptatietester, welke vaak nog Excel-sheets gebruikt voor het uitvoeren van zijn taak, wordt met Visual Studio 2010 ruimschoots ondersteund. Testcases worden geregistreerd in TFS en



TESTPLAN OVERZICHT IN CAMANO VAN WAARUIT TESTCASES WORDEN UITGEVOERD



IMPACT ANALYSE SCHERM

hij/zij krijgt een specifiek hulpmiddel dat gebaseerd is op het optimaal ondersteunen van de doorgaans handmatige testwerkzaamheden.

Hoe ziet dit er dan allemaal uit en waar wordt de samenwerking daadwerkelijk ondersteund? In de volgende paragrafen worden aan de hand van de bovenstaande testdoelen en testtaken de capaciteiten van Visual Studio Team System 2010 nader uitgelegd.

Democratizing Application Lifecycle Management

Met de visie 'Democratizing Application Lifecycle Management' van Microsoft is de ondersteuning die Visual Studio Team System 2010 biedt op het gebied van ALM verder uitgebreid. Er zijn vele toevoegingen en extra hulpmiddelen beschikbaar om beter samen te werken en de verschillende rollen worden beter ondersteund.
Er zijn verschillende technieken en componenten op het gebied van testen aan Visual Studio toegevoegd. De meest in het oog springende toevoeging is de stand-alone [acceptatie]testtool Camano. Camano is een testmanagementomgeving waarbinnen testcases aangemaakt kunnen worden, die workitems van een nieuw workitemtype 'testcase' binnen Team Foundation Server zijn. Testcases kunnen worden uitgevoerd met de TestRunner, die de mogelijkheid biedt om de acties van de tester op te nemen en mee te zenden met een bevindingenrapport. De TestRunner neemt niet alleen het scherm van de tester op, maar verzamelt ook informatie over het systeem waarop getest wordt en maakt een bestand aan dat gebruikt kan worden om de acties van de tester in debug-modus te doorlopen binnen Visual Studio. Dit bestand is in feite een complete stacktrace waarmee de debug ervaring is alsof je real-time aan het debuggen bent. Microsoft noemt dit 'Historical debugging'.

De Visual Studio Test Edition biedt naast deze ondersteuning voor het uitvoeren van de acceptatietests ook de mogelijkheid om ze te automatiseren. Met behulp van CodedUI technologie welke

C# code genereert uit de acties die in het scherm zijn uitgevoerd kunnen UI tests meegenomen worden in het dagelijkse of continuous buildprocess.
Veel nieuwe dingen, maar hoe gebruik je ze?

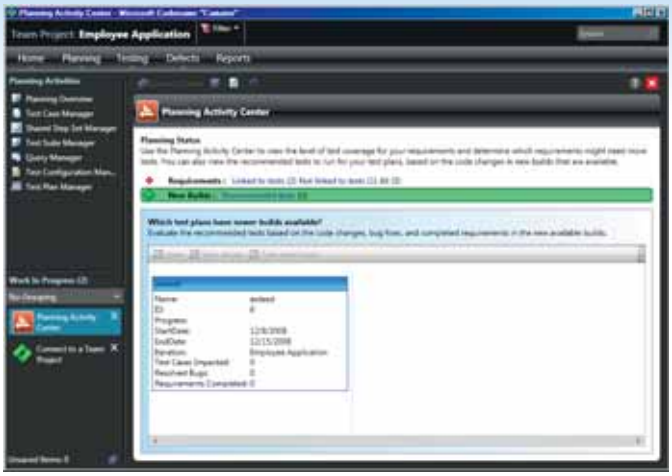
Test Impact View voor de ontwikkeltester

De ontwikkeltester is in de voorgaande en huidige versie van Visual Studio al goed voorzien van hulpmiddelen om zijn/haar taken uit te voeren. Zo is er het testframework en zijn er testdekkingrapportages om de kwaliteit aan te tonen.
Microsoft heeft toch nog iets gevonden om het werk van de ontwikkeltesters en degenen die Test Driven Development doen helpen. De Test Impact Analyse is hiervan de meest aansprekende.

De Test Impact View geeft ontwikkelaars (testontwikkelaars) na een codewijziging een overzicht van alle unit tests die beïnvloed zijn en dus opnieuw uitgevoerd dienen te worden. Naast de tests die uitgevoerd dienen te worden, geeft de test impact view ook een overzicht van de code die is aangepast. Dit overzicht zorgt er dan ook voor dat code changes efficiënt worden getest.

Camano, TestRunner en CodedUI voor de systeem- en acceptatie testers

Acceptatietests verifiëren het systeem op het gebied van inpasbaarheid en functionaliteit, zoals hiervoor is beschreven. Degenen die de rol van acceptatietester uitvoeren kunnen dit volledig 'disconnected' van het ontwikkelteam. Het enige wat nodig is, is een lijst met tests die uitgevoerd dient te worden. Wanneer de applicatie niet doet wat in de test is beschreven, wordt deze opgestuurd naar de ontwikkelaar die het dan kan gaan oplossen.

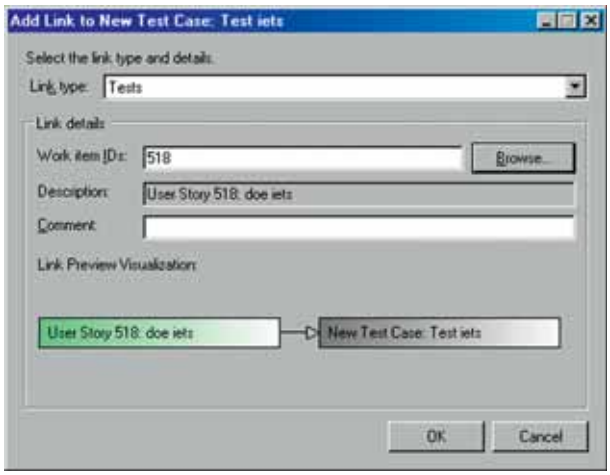


TESTIMPACT CAMANO, USERSTORIES ZONDER TESTEN

Er zijn drie uitdagingen in dit eenvoudige scenario:

De eerste: testcases zijn gebaseerd op eisen vanuit de business. Negen van de tien keer zijn deze eisen in het begin van het project opgezet door de business en gaande de ontwikkeling van de applicatie aangepast door voortschrijdend inzicht. Hierdoor ontstaat een probleem voor de tester. Hoe weet hij dat de testcase die hij uitvoert up to date is en terug te herleiden is tot een gebruikerseis. Binnen Visual Studio 2010 is deze uitdaging het hoofd te bieden met de mogelijkheid om testcases te koppelen aan User Stories (in de Agile Template) of Requirements (in de CMM processtemplate). Daardoor kunnen rapportages gemaakt worden over deze

Ook een video van de handelingen die de test heeft uitgevoerd kan worden opgeslagen bij het bevindingenreport



TESTCASE KOPPELEN AAN EEN USERSTORY

Userstories. Binnen Camano, het van Visual Studio losstaand hulpmiddel voor de acceptatietester, wordt inzichtelijk gemaakt wanneer er Userstories zijn die nog geen testcase hebben en andersom. Ook wordt getoond wanneer een testcase niet gekoppeld is aan een Userstory, dit zou dan een overbodige test kunnen zijn.

Net als bij de test 'Impact Analyze' binnen Visual Studio heeft Camano ook een impact view die aangeeft welke tests opnieuw uitgevoerd dienen te worden door aanpassingen in code, maar ook door aanpassingen in Userstories. Hiermee de tester helpend beter inzicht te geven in de werkzaamheden van de business analyse (en ontwikkelaar), door aan te geven dat er iets veranderd is, wat er veranderd is en dat hij iets moet aanpassen cq. een test opnieuw moet uitvoeren.

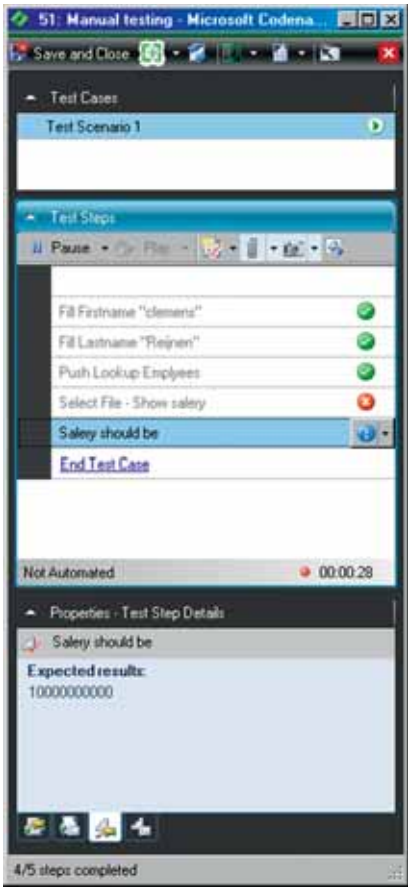
Het tweede probleem zijn de bevindingen. De tester vindt iets wat niet klopt in de applicatie en stuurt deze bevinding in een mailtje of ander meer geavanceerd bug-reporting systeem met prio-high naar de ontwikkelafdeling. De ontwikkelaar die verantwoordelijk gemaakt is voor deze bug pakt hem vervolgens op, probeert hem te reproduceren en alles blijkt gewoon te werken. Mailtje terug naar de tester met de status aanpassing: 'bug solved' of 'not reproduceable'. De tester controleert het, ziet dat de bug er nog steeds in zit en stuurt hem weer, iets geïrriteerder, terug. Het begin van het zogenaamde bug ping-pongen.

Vanaf de eerste release van Team Foundation Server in 2005 is de 'bug' al als workitem onderkend Dit maakte het voor ontwikkelaars eenvoudiger en eenduidiger om bugs te registeren, te onderhouden en rapportages te maken. Maar van echte integratie met acceptatietesters is geen sprake. Deze kunnen bevindingen opvoeren door gebruik te maken van de WebAccess tool van TFS, maar echt los je daar de bovenstaande uitdagingen niet mee op.

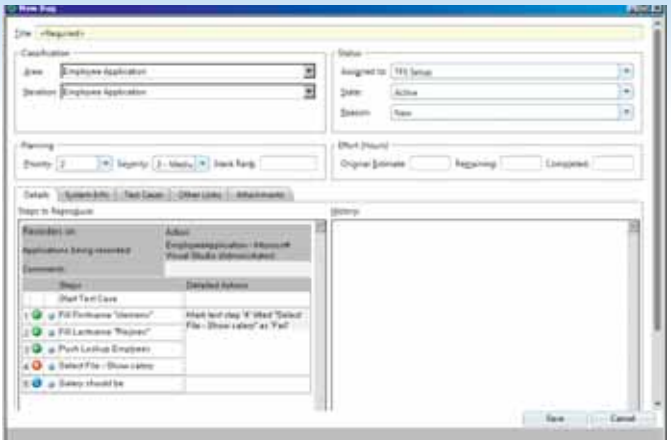
Met Visual Studio 2010 en de stand alone testtool Camano wordt deze integratie met acceptatietesters werkelijkheid. Met de Test-Runner van Camano en wordt het mogelijk om binnen de bevindingenregistratie veel benodigde informatie op te slaan die van nut kan zijn voor de ontwikkelaar om de bevinding te reproduceren en op te lossen.

Microsoft Testrunner maakt een snapshot van de omgeving, operating systeem versie, service pack en alle mogelijke systeemgegevens. Naast deze systeemgegevens kan de tester tijdens het uitvoeren van de test screenshots maken en inbedden in het bugreport. Niet alleen schermfoto's kunnen genomen worden maar ook een video van de handelingen die de test heeft uitgevoerd kan worden opgeslagen bij het bevindingenreport, dat geïndexeerd wordt naar de teststappen in de testcase. Al deze extra informatie helpt de ontwikkelaar om sneller en eenvoudiger de oplossing te vinden en zorgt er ook voor dat de muur, de irritatie, tussen ontwikkelaars en testers kleiner wordt, zo niet verdwijnt.

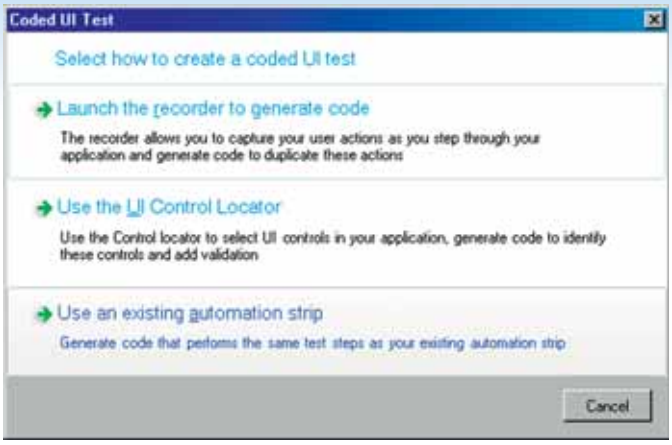
De derde uitdaging is de werkvoorraad van de tester. Wanneer bijvoorbeeld een bevinding is opgelost, dient hij zijn testcase opnieuw uit te voeren. Dit is op zich geen grote uitdaging, omdat hij alles in Excel of ander hulpmiddel heeft vastgelegd. Maar wat als bijvoorbeeld de bevinding een onderdeel is dat op honderden pagina's of schermen gebruikt wordt? Dan moeten al deze pagina's



TESTRUNNER



BEVINDINGEN WORKITEM MET DE TESTSTAPPEN IN DE DETAILS-TAB



VERSCHILLENDE MOGELIJKHEDEN OM UI TESTS TOE TE VOEGEN AAN TESTPROJECTEN. DE 'AUTOMATION STRIP' OPTIE KAN STRIPS HERGEBRUIKEN VAN CAMANO

opnieuw getest worden wat niet heel effectief is voor de tester. Camano maakt het mogelijk om TestCases op te nemen en deze naderhand weer af te spelen binnen de TestRunner, waarbij de tester dus niet handmatig alle stappen meer hoeft te doorlopen. Een sterke verbetering in het verminderen van herhalende werkzaamheden. Een andere toevoeging aan Visual Studio welke het leven van de acceptatie, systeemtester eenvoudiger maakt, is de CodedUI. Hiermee wordt het mogelijk om scripts te maken voor de moeilijk automatisch te testen user interface.

De code die nodig is om deze tests mogelijk te maken kan geëxtraheerd worden uit de tests die uitgevoerd zijn binnen Camano. Slechts kleine aanpassingen hoeven plaats te vinden om ze ook daadwerkelijk automatisch te laten uitvoeren (met variabele data bijvoorbeeld). Hiermee kunnen vervolgens loadtests, regressietests en gebruikerstests geautomatiseerd worden, waardoor het voorbeeld van de honderden te hertesten pagina's voorkomen kan worden.

Gestructureerd testen

Testtools brengen geen structuur in het testproces, ze maken de werkzaamheden van de tester makkelijker. Sterker nog, het automatiseren van ongestructureerd testen werkt CHAOS-verhogend! Binnen Camano is het nog steeds mogelijk om vele onzinnige testcases aan te maken. Hoe ziet een gestructureerde testaanpak er dan uit? Hiervoor zijn vele invullingen te bedenken.

EEN ORGANISATIE DIENT TE KIEZEN MET WELK AMBITIENIVEAU DIT GESTRUCTUREERDE TESTPROCES WORDT GEÏMPLEMENTEERD IN DE PROJECTEN

MET DEZE 'DEFINITIE VAN KLAAR' IS HET MOGELIJK OM EEN LEERLUS IN TE RICHTEN: DUS NIET MEER ALLEEN DE FOUTEN HERSTELLEN, MAAR OOK LEREN HOE ZE VOORKOMEN KUNNEN WORDEN!

In het algemeen kan worden gezegd dat een gestructureerde testaanpak wordt gekenmerkt door:

- Het bieden van structuur, zodat duidelijk is wat, door wie, wanneer en in welke volgorde moet worden gedaan.
- Het omvatten van de volledige scope en de beschrijving van het complete scala aan relevante aspecten.
- Het bieden van concrete handvatten, zodat niet steeds opnieuw het wiel uitgevonden hoeft te worden.
- Het sturen van de testactiviteiten in het kader van tijd, geld en kwaliteit.

Binnen Visual Studio Team System 2010 is het mogelijk om een procestemplate voor gestructureerd testen in TFS op te nemen.

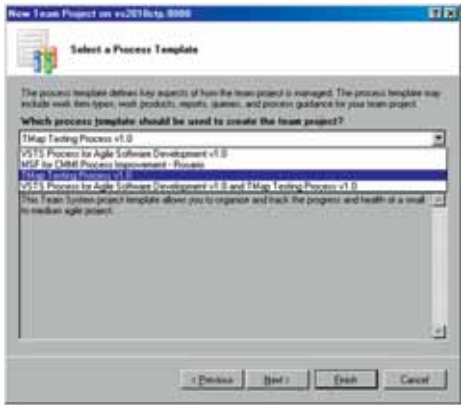
Sogeti werkt aan twee versies van de 'TMap® proces template':
1) De stand alone versie, toepasbaar voor de acceptatie testers.
2) En de merge versie, die 'achter' de door het ontwikkelteam gebruikte proces template geplakt wordt.

Toch is alleen (gestructureerd) testen niet voldoende. Als een ontwikkelaar namelijk niet weet welke kwaliteit geleverd dient te worden, weet hij ook niet wanneer hij klaar is. De tester weet niet wanneer een fout wel of niet door de ontwikkelaar zelf gevonden had moeten worden.
Het is dus belangrijk om de te leveren kwaliteit af te spreken. In TMap® Next (de gestructureerde testaanpak van Sogeti) is hiervoor het begrip basiskwaliteit geïntroduceerd. Deze basiskwaliteit beschrijft niet alleen de benodigde testdekking, maar geeft ook handvatten voor de wijze waarop de tests dienen plaats te vinden, en beschrijft de benodigde bewijsvoering en naleving.

Gestructureerd testen, het efficiënt maken van het volledige testproces, zodat testers hun werkzaamheden doen op basis van risico, (basis)kwaliteit en leerlussen is iets wat met Visual Studio Team System 2010 te realiseren is. Afspraken over procedures, basiskwaliteit en andere dienen hun weg te vinden in de procestemplates van Team Foundation Server.

Conclusie

Met Visual Studio Team System 2010 krijgen alle testrollen een duidelijke en betere ondersteuning binnen de Application Lifecycle. Testers staan niet meer alleen bij het gebruik van technische hulpmiddelen, maar zijn geïntegreerd in de hulpmiddelen die architectuur en ontwikkelaars ook gebruiken, hiermee de muur slechtend. Maar goede hulpmiddelen alleen zijn niet genoeg. Daarnaast is een duidelijke onderscheiding van rollen, taken en bevoegdheden nodig. Als laatste en belangrijkste bepaalt een gestructureerde manier van gebruik of je succesvol bent met je teststrategie.



TMAP PROCESTEMPLATE VOOR GESTRUCTUREERD TESTEN IN VSTS 2010

Links

http://www.codeplex.com/TestingGuidance
www.TMap.net
www.robkuijt.nl
www.clemensreijnen.nl
Information on VSTS2008 Test Edition:
http://msdn.microsoft.com/en-us/vsts2008/products/bb933754.aspx
Information on VSTS2010 Test Edition (Rosario):
http://msdn.microsoft.com/en-us/vsts2008/bb725993.aspx

Rob Kuijt is als Senior Testconsultant werkzaam bij Sogeti (www.sogeti.nl).
Clemens Reijnen is MVP Team System en werkzaam bij Sogeti.

Meer weten is meer KUNNEN.

motion10 heeft meer BizTalk Server en SharePoint implementaties in meer landen gedaan dan welke andere integrator ook. Als team kunnen we alles aan!

motion10

The reliable specialist in system integration

Implementatie – Ondersteuning – Outsourcing

www.motion10.com **info@motion10.com**

010-2351035 Rivium Quadrant 151 2909 LC Capelle a.d. IJssel Nederland

Microsoft GOLD CERTIFIED Partner