PDC 2008 en Visual Studio Team System 2010

Significante stap voorwaarts

Marcel de Vries

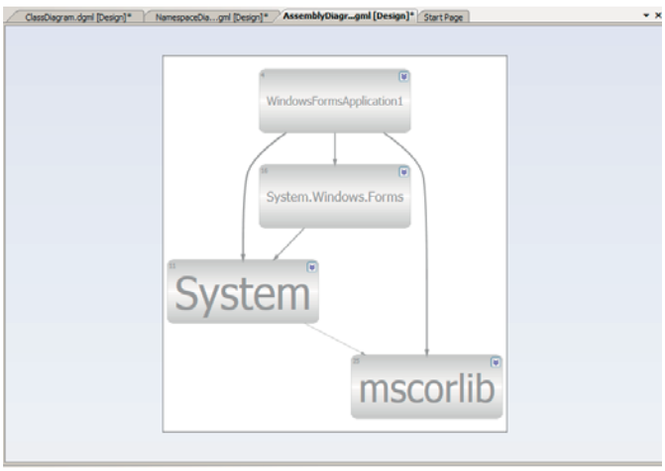
Op de Professional Developers Conference hebben we de beschikking gekregen over een nieuwe Community Technical Preview van Visual Studio Team System 2010. Bij een eerste blik op deze CTP zal je overweldigd raken door alle nieuwe onderdelen. In dit artikel een overzicht van de meest in het oog springende nieuwe features in de Visual Studio Team Edities (Architect, Development en Test) en de Team Foundation server.

In de 2010-versie zijn Team Development en Team Database Edition samengevoegd tot één team-editie. Sinds oktober dit jaar is dit licentietechnisch ook al het geval voor de 2008-versie. In de 2010-versie worden de producten ook volledig samen geleverd als één.
Binnen de Team Development Edition heeft men zich voornamelijk beziggehouden met een belangrijk probleem bij het oplossen van geconstateerde bugs. De ontwikkelaar kan deze bugs dan niet reproduceren op zijn ontwikkelomgeving. Deze zogenoemde 'non repro' bugs treden voornamelijk op in situaties met verschillen tussen ontwikkelomgeving en test- of productie-server. Er komen gevallen voor waarin het echt onmogelijk is het probleem te reproduceren. Dit wordt in steeds grotere mate veroorzaakt door verschillen in processors, netwerkverbindingen, beveiliging, et cetera. Zeker met de komst van MultiCore-processors en de groeiende complexiteit van software valt het te verwachten dat dit probleem in de toekomst alleen maar gaat toenemen. Om het non-repro-probleem het hoofd te bieden, introduceert Microsoft een belangrijke nieuwe innovatie onder de naam historical debugger.
De historical debugger kun je vergelijken met de flight data-recorder in een vliegtuig. Bij een probleem (in het ergste geval een crash) is alle data beschikbaar om een analyse te maken van de oorzaak. Historical debugging werkt in essentie hetzelfde. Tijdens het testen van een applicatie (of eventueel zelfs in een productieomgeving) draait deze 'datarecorder' op de achtergrond mee. Hij slaat alle belangrijke interacties van de applicatie met zijn omgeving op. Denk aan registry-toegang, File IO, Netwerk IO, Windows Event traces, et cetera. Bij een probleem in de software koppelt de tester na het rapporteren van een bug de historische debug-log aan het bug-workitem. Deze log laadt hij vervolgens via Visual Studio in de debugger. Daarna kan de ontwikkelaar door de software heen stappen alsof het systeem op zijn machine werd uitgevoerd. Hij ziet waarden van variabelen, bekijkt de call-stack en krijgt inzage of er exceptions in de applicatie zijn opgetreden. Hij stapt door de code heen op exact dezelfde manier als hij dat zou doen bij het debuggen van een live-programma. De integratie gaat zelfs zover dat de debugger kan switchen tussen live mode en historical mode. Bijvoorbeeld als de ontwikkelaar live aan het debuggen is en ook de historical debugger actief heeft. Dit betekent een zeer grote vooruitgang om het no-repro-probleem het hoofd te bieden. Als ontwikkelaar hoef je niet eerst veel energie te steken in het reproduceren van het probleem, voordat je het kunt opsporen. De integratie met workitems maakt dit helemaal zeer krachtig. Door op het attachement in het bug-workitem te klikken, activeer je direct de historical debugger.

Test Impact Analysis

Naast historical debuggen is ook gekeken naar het concept van Impact Analysis. Daarbij heeft men zich eerst gefocust op de vraag welke unittesten uitgevoerd moeten worden ten gevolge van het aanpassen van code. Tijdens het aanbrengen van aanpassingen in de code kan de omgeving de minimaal uit te voeren set unittesten bepalen om te valideren of een aanpassing mogelijk regressie tot gevolg heeft gehad. Het idee daarachter is dat er vaak enkele tientallen en soms zelfs honderden unittesten ter beschikking staan, maar men geen idee heeft welke testen kunnen aantonen of een aanpassing tot regressie heeft geleid. Test Impact Analysis bepaalt op basis van onder andere de code-coverage-data uit eerder uitgevoerde unittesten of een unittest noodzakelijk is. Bij constatering dat een unittest bepaalde regels zojuist aangepaste code zou raken, wordt deze unittest aangemerkt als 'aanbevolen'. Na afloop van het doorvoeren van je verandering kan je deze aanbevolen lijst uitvoeren en controleren of een van de testen faalt.
Impact Analysis is overigens niet alleen erg handig in de IDE, maar ook juist tijdens het uitvoeren van een Build. Zeker in het geval je zogenaamde 'Continious Integration' (CI) build op een project hebt aangezet. CI heeft namelijk tot doel een build te

> Tijdens het testen van een applicatie draait de 'datarecorder' op de achtergrond mee



ILLUSTRATIE 1. ASSEMBLY DEPENDENCY DEFAULT WINDOWS FORMS-APPLICATIE

maken die snel klaar is, zodat we deze vaak kunnen uitvoeren. Deze moet wel direct aantonen dan een change mogelijk problemen veroorzaakt in de totale codebase. Door nu in een CI build alleen de 'aanbevolen' testen uit te voeren in plaats van de hele set, gebruiken we een minimale set unit-testen voor het aantonen van regressie in de build. Dit stelt ons in staat een CI build zeer efficiënt in te richten.

Visual Studio Team Architect

Binnen Team Architect heeft men zich primair gefocust op het inzichtelijk maken van de architectuur van een product. Met als uitgangspunt dat er feitelijk twee stromingen bestaan in softwareontwikkeling. Bij de eerste, Agile, wordt relatief weinig ontwerp toegepast, maar wil men wel inzicht hebben in de structuur van de programmatuur en de bijbehorende kwaliteit. De tweede stroming is meer formeel, waarbij we vanuit een topdown-design werken. In het Agile-scenario beginnen we meestal niet met een ontwerpdiagram vooraf, maar juist met het schrijven van test cases die helpen bij de implementatie van de functionaliteit.

Architecture explorer

Ter ondersteuning van deze bottom-up benadering dient de zogenaamde architecture explorer. Met deze tool kunnen we bestaande code op verschillende manieren visualiseren. Een van die manieren is het tonen van de structuur, in de zin van afhankelijkheden op het gebied van assemblies, namespaces en classes. Een dergelijk diagram biedt inzicht in softwareonderdelen met beperkte mogelijkheden qua onderhoud. Bijvoorbeeld door de afhankelijkheid van een bepaalde class. Dit soort diagrammen zijn bedoeld om mogelijke problemen te ontdekken. Illustratie 1 bestaat uit een assembly dependency-visualisatie van een simpele applicatie, standaard aangemaakt in een Windows Forms-applicatie van Visual Studio .

Ondersteuning voor UML

Microsoft heeft er (eindelijk) voor gekozen in Team Architect ondersteuning te bieden aan UML 2.1. Sterker nog, Microsoft is ook toegetreden tot het voor UML-specificaties verantwoordelijke OMG. Microsoft draagt op deze manier bij aan de verdere ontwikkeling van de UML-specificatie. In de 2010-release voegt men in Team Architect de meest gebruikte diagrammen voor software-engineering toe. Op dit moment zijn de volgende diagrammen terug te vinden in de CTP: (conceptual) Class Diagram, (conceptual) Sequence Diagram, Activity Diagram, Component Diagram en het Use Case Model. Daar waar conceptual tussen haakjes staat is het zowel mogelijk conceptuele diagrammen te maken, die bijvoorbeeld een architect in de beginfase van een project zal produceren, als diagrammen die een alternatieve kijk op de broncode weergeven, dus gerelateerd aan de daadwerkelijke implementatie.

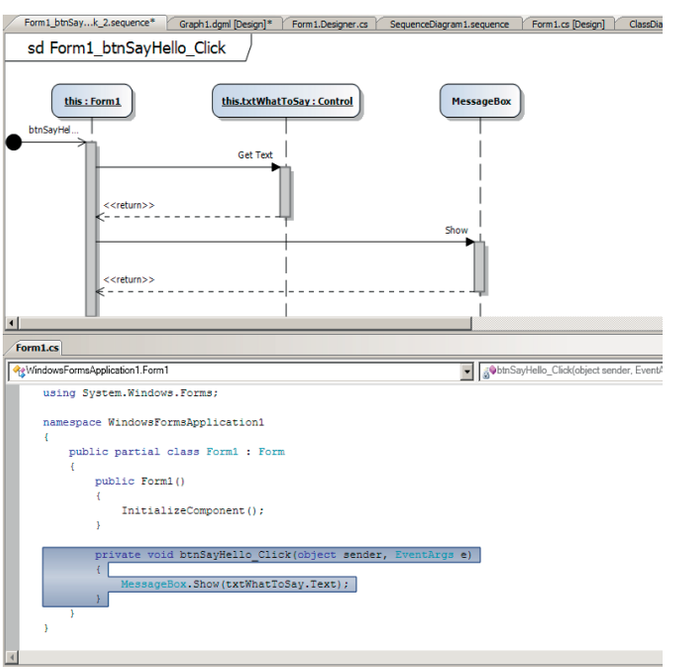
Verder is een nieuw (niet-UML) diagram geïntroduceerd, te weten het Layer Diagram. Daar kan een architect in aangeven welke logische lagen in de architectuur aanwezig zijn en welke regels gelden voor communicatie tussen deze layers. Denk aan bijvoorbeeld een klassieke indeling in drie lagen: UI, Business en Data. Het mooie is dat aan de hand hiervan ook direct validatie kan plaatsvinden op de source code. Op die manier kunnen we alle geschreven code valideren tegen de architecturale lagen. Overtredingen tegen de regels worden aangemerkt als een compilatiefout. Hierdoor vindt vroegtijdig terugkoppeling plaats zodra iemand tegen de afgesproken regels in software incheckt in versiebeheer.

In de CTP ontstaat al een goede indruk hoe de uiteindelijke diagrammen gaan werken. Krachtig aan de implementatie van het Sequence Diagram is bijvoorbeeld dat men goed heeft nagedacht over het snel vol raken van een diagram, met het gevaar dat je het overzicht verliest. Door het eenvoudig in- en uitklappen van delen van een sequence behoud je het overzicht. Een andere nieuwe opties helpt om delen van een sequence te verplaatsen naar een ander diagram, terwijl dit via een 'click through' eenvoudig benaderbaar blijft. Zo kan een architect zijn intenties duidelijk maken, waarna de ontwikkelaar de detailinvulling completeert.

Illustratie 2 bestaat uit een screenshot van een gegenereerd Sequence Diagram. Dit diagram is gegenereerd door in de code rechts te klikken en te kiezen voor de optie 'Generate Sequence Diagram'.

Visual Studio Team Test

Bij de huidige versie (2005 en 2008) van de Team Test Editie is de doelgroep heel duidelijk de 'technische' tester. De 2010-versie heeft tevens de functionele tester in het vizier. Deze tester krijgt een geheel nieuwe toolset die buiten de Visual Studio IDE werkt. De tool draagt op dit moment nog de naam 'Camano'. Het gaat



ILLUSTRATIE 2. SEQUENCE DIAGRAM GEGENEREERD VOOR DE GESELECTEERDE CODE

om een (in WPF geschreven) applicatie die ondersteuning biedt in het opzetten, beheren, plannen en uitvoeren van functionele testen. Camano maakt gebruik van een aantal nieuwe termen die overal in de tooling terugkomen, te weten Test Case, Test Suite, Test Configuration, Test Plan en Test Pass. In het kort staan deze termen voor het volgende:

- Een Test Case bevat de stappen voor een specifiek onderdeel van een te testen applicatie. Een Test Case zelf wordt als een workitem opgeslagen in de Team Foundation Server met onder andere de in een test uit te voeren stappen.
- Een Test Suite is een samenvoeging van een aantal Test Cases. Test Cases selecteer je uit een lijst of door middel van een query. Dit laatste noemen we een Dynamic Test Suite.
- Test Configuraties zijn aanduidingen van omgevingen waar een test wordt uitgevoerd. Bijvoorbeeld Windows Vista SP1 met FireFox browser, of Windows XP SP3 met IE7.
- Een Test Plan is een samenvoeging van Test Suites en Test Configurations. In een Test Plan kun je specifieke testen toekennen aan testers en het Test Plan activeren zodat automatisch de benodigde workitems op de naam van een betreffende tester komen te staan.
- Het uitvoeren van een test noemen we een Test Pass. Op een Test Pass worden de resultaten van een test gerapporteerd.

Eén van de hulpmiddelen is een video-opname om te registreren wat de tester allemaal heeft uitgevoerd

De tester geeft de resultaten tijdens het uitvoeren van de test aan in het test tool 'MSRun'. Deze test tool dient ter ondersteuning van de tester. De tool biedt een aantal opties, waaronder het op de achtergrond opnemen van uitgevoerde acties, vastleggen van de machine-instellingen (zoals Capslock, Numlock en Service packs), opnemen van een zogenaamde automation strip en activeren van de historical debugging log.
Al deze opties dragen bij aan het zoveel mogelijk vergaren van informatie die de ontwikkelaar helpt de gevonden problemen snel duidelijk te maken. Eén van de hulpmiddelen is een video-opname om te registreren wat de tester allemaal heeft uitgevoerd. Deze video wordt opgeslagen bij het rapporteren van een Bug, zodat de ontwikkelaar direct kan zien waar het probleem zich heeft voorgedaan. Ook de historical debug logs worden opgeslagen. Hierdoor kan de ontwikkelaar gelijk met de historical debugger door de historische data van de testrun stappen om het probleem snel boven tafel te krijgen.

Test automation

Bij het uitvoeren van de test is er een optie om een automation strip op te nemen. Deze bevat alle data om het uitgevoerde scenario volledig automatisch opnieuw te laten afspelen via een test automation engine. De automation kunnen we voor twee doeleinden inzetten. Ten eerste voor de ondersteuning van een hertest, waarbij een tester bijvoorbeeld de eerste dertig stappen uit het scenario automatisch laat uitvoeren, en vervolgens zelf alleen de laatste stappen die dienen ter verificatie of de bug is opgelost. De tweede optie is het automatiseren van de test als een zoge-

naamde 'Coded UI' Test. Zo'n test is voornamelijk bedoeld om de eerder uitgevoerde testen volledig te automatiseren en onderdeel te maken van je build. Houd er rekeningen mee dat dit voornamelijk nuttig is na oplevering en stabiele werking van een product. Anders moet je continu de testcode aanpassen. Met de Coded UI-test is relatief eenvoudig regressie in de code, door bijvoorbeeld een bugfix, aan te tonen. Door dit in de build te integreren kun je de nazorg van een applicatie veel beter en goedkoper onder controle houden.

Bij het maken van een Coded UI-test dient de automation strip als basis. De automation strip genereert namelijk code die alle stappen volgt zoals die ook door de tester zijn uitgevoerd in een eerdere Test Pass. Na het genereren van de code moeten we verificatiepunten toevoegen aan de code. Deze valideren of bepaalde gewenste resultaten daadwerkelijk aanwezig zijn. Toevoegen van een verificatiepunt aan de code gaat eenvoudig met tools in Visual Studio. Bij een verificatiepunt kan je testen op allerlei in de UI verwachte onderdelen, zoals de tekst in een control en de kleur van een achtergrond.

Ook de rapportage over de uitgevoerde Test Pass(es) is volledig in Camano beschikbaar en geeft een grafisch overzicht van de testresultaten voor een betreffend testplan. Camano biedt door middel van koppeling met Team Foundation Server de mogelijkheid in het team samen te werken. De testcases, plans en results worden allemaal opgeslagen in Workitem Tracking en voor wat betreft de testresultaten en voortgang in een aparte Team Test Database. Illustratie 3 bestaat uit een screenshot van Camano met de resultaten van een Test Pass.

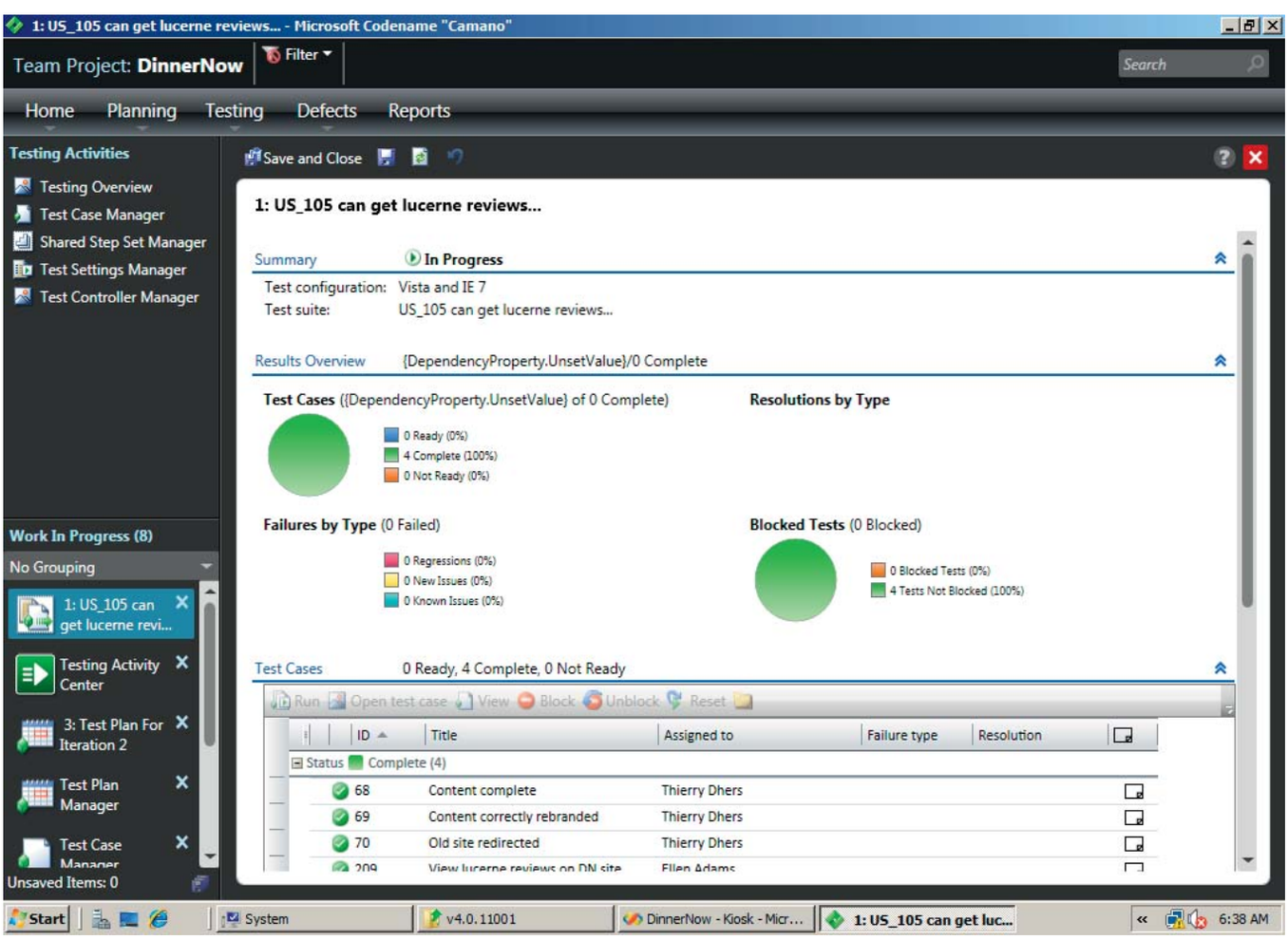
Team Lab

Volledig nieuw in de 2010-omgeving is Team Lab voor onderhouden en opzetten van een virtuele op Hyper-V-technologie gebaseerde testomgeving. Met als idee dat je na een build ook de testen zo snel mogelijk wilt uitvoeren. Om dit mogelijk te maken, bestaat er behoefte aan het geautomatiseerd installeren van de laatste build op een schone testomgeving. Virtualisatietechnologie zoals Hyper-V is hiervoor uitermate geschikt, omdat we bijvoorbeeld een server eenvoudig kunnen terugzetten naar zijn originele status. Na het vinden van een bug is het eenvoudig een snapshot van de server te maken. Zo'n snapshot kunnen we vervolgens aan de ontwikkelaar aanbieden als troubleshoot-omgeving. Op dit moment is Team Lab aangekondigd, maar de volledige feature set van dit onderdeel is nog niet helemaal helder. Hierover zal in de komende periode snel duidelijkheid komen en in een komend nummers van .NET Magazine gaan we hier verder op in.

> Het selecteren van een machine uit de pool gebeurt op basis van tags

Team Foundation Server

Team Foundation Server bestaat primair uit drie onderdelen, te weten Version Control, Workitem Tracking en Team Build. Op al deze onderdelen zijn drastische verbeteringen doorgevoerd. Op



ILLUSTRATIE 3. DETAILS VAN CAMANO TEST PASS

het gebied van workitem tracking behoren onder andere hiërarchische workitems tot de mogelijkheden. In het verleden konden we wel workitems aan elkaar linken. Maar hierbij was geen duidelijke parent child-relatie zichtbaar in onder andere de resultatenlijstjes in Visual Studio, Excel en Project. Naast het vastleggen van een workitem-hiërarchie is het ook mogelijk queries uit te voeren om te kijken of er een bepaalde relatie tussen workitems bestaat. Een voorbeeld daarvan: het bepalen welke user scenario workitems nog niet aan een testcase workitem is gelinkt. Dit om te bepalen voor welke scenario's we nog een testcase moeten maken.

Version control

Voor version control is hard gewerkt om het branchen en mergen beter inzichtelijk te maken. Zo is een branch nu een apart 'object', gevisualiseerd in de source control explorer. Verder heeft men de historie van een changeset beter inzichtelijk gemaakt in relatie met de aanwezige branches. Momenteel is niet altijd duidelijk of een bepaalde changeset nu wel of niet onderdeel uitmaakt van een specifieke branch. We kunnen nu visualiseren in welke branch een changeset al is doorgevoerd. Zeer nuttig bijvoorbeeld om inzichtelijk te maken in welke versie van een product een Bugfix is doorgevoerd. Illustratie 4 geeft hier een beeld van. De visualisatie toont dat changeset 27 is ontstaan vanuit een check in branch serviceportfolio met changeset nummer 26 en vervolgens terecht is gekomen in de branches Integration; Test en Delivery.

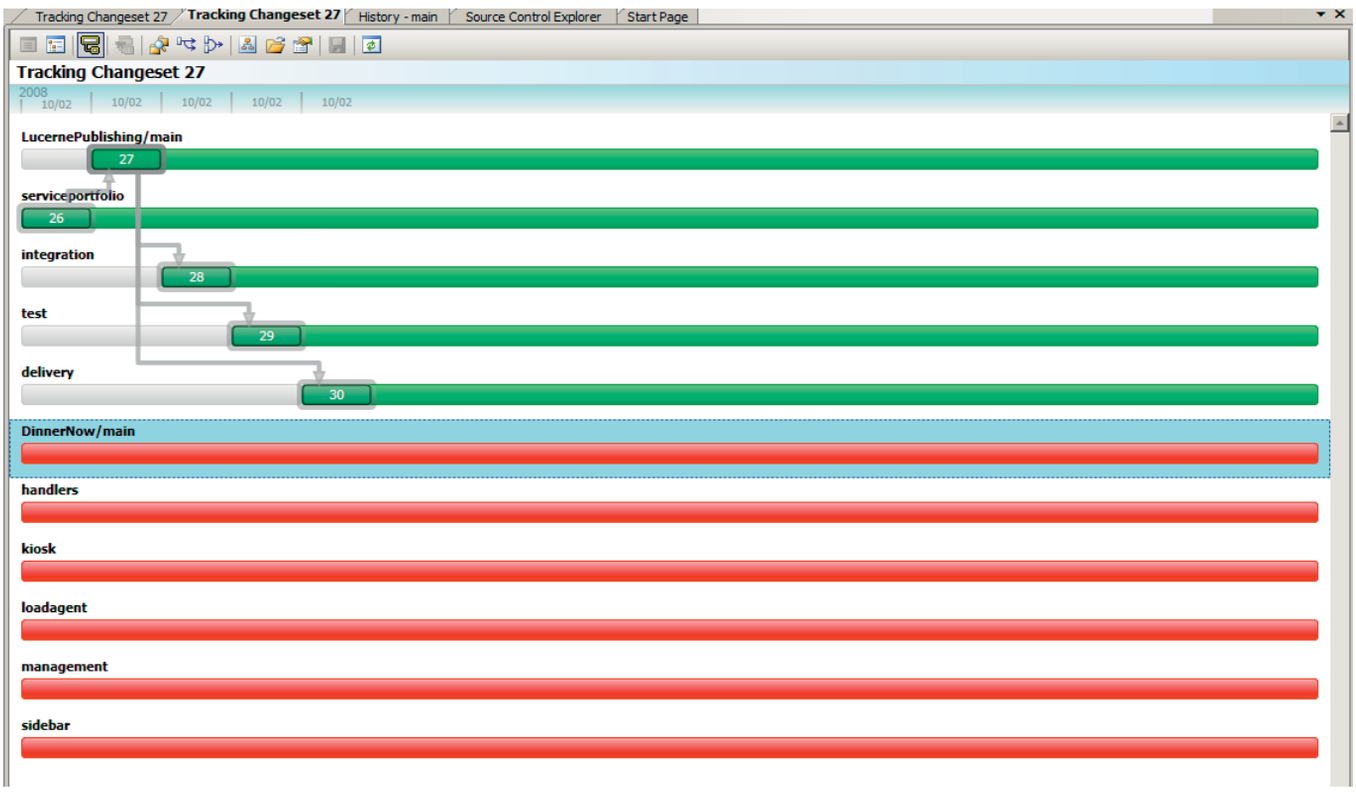
Met WF kan men naast de eenvoudige build-visualisatie ook een veel robuustere runtime-build-omgeving realiseren

Naast deze changeset tracking kunnen we nu ook de branch-hiërarchie visualiseren. Daarbij is naast de standaard folder view, die we al gewend waren, ook een branch view beschikbaar. Dit soort visualisatie is vooral heel nuttig bij gebruik van een complexe branch-structuur.

Team Build

Ten slotte zijn er aanpassingen gemaakt in de team build-omgeving. In de 2008-versie kwam al een aardige aanpassing in de architectuur. Toch heeft men er voor gekozen ook in 2010 een significante verandering in de architectuur door te voeren. Een van de veelgevraagde mogelijkheden is namelijk de optie om niet een eenmalig MSBuild-script te genereren voor de build, maar deze continu aanpasbaar te houden vanuit de IDE. Daarbij is ook betere visualisatie gewenst. Mede om die reden heeft het team ervoor gekozen de toekomstige Team Build-omgeving weer helemaal opnieuw te bouwen.

De nieuwe omgeving is nu volledig gebaseerd op Windows Workflow Foundation (WF). Door middel van WF kan men namelijk naast de eenvoudige build-visualisatie (Workflow designers) ook een veel robuustere runtime-build-omgeving realiseren. WF biedt standaard ondersteuning voor zaken als parallel uitvoeren



ILLUSTRATIE 4. TRACK CHANGESET-VISUALISATIE

van stappen en het distribueren van stappen over machines heen. Last but not least geeft dit ook de oplossing voor het aanpasbaar houden van de build-stappen door middel van een editor. Verder heeft men nagedacht over het beter gebruikmaken van een set build servers. Het is nu mogelijk een build uit te laten voeren op een vrije build-machine uit een pool met machines. Hierbij vindt de afweging plaats of een build server uit de pool voldoet aan gestelde randvoorwaarden, alvorens deze in te schakelen om de build uit te voeren. Het selecteren van een machine uit de pool gebeurt op basis van tags. Een tag is feitelijk niet meer dan een eigenschap die je aan een build server en een build-definitie hangt. Als een gelijke set aan tags in de build server wordt gevonden, zoals aangegeven in de build-definitie, geeft dit aan dat een server geschikt is voor het uitvoeren van deze build. Op basis daarvan volgt selectie van de machine.

Een grote sprong voorwaarts op het gebied van Application Lifecycle Management

Conclusie

Al met al kunnen we stellen dat Team System 2010 een significante stap voorwaarts is. Als je de CTP vanaf de MSDN-website downloadt, is daarop een zeer uitgebreide set 'walk througs' beschikbaar. Deze bieden op alle genoemde features een tutorial om de werkende onderdelen van een feature stap voor stap te doorlopen. Erg handig als je in weinig tijd alle nieuwe onderdelen zelf wilt uitproberen. Zoals in de introductie aangegeven, biedt dit artikel een overzicht van wat er allemaal aan zit te komen in de 2010-versie. Begin volgend jaar zal een themanummer verschijnen met diverse artikelen die zijn toegespitst op de specifieke features waarbij er in veel meer detail naar wordt gekeken. Lees ook het interview in deze editie met James Whitaker en Neelesh Kamkolkar over Test features en Team Lab.

Op basis van wat ik tot op heden heb gezien, ben ik van mening dat Visual Studio Team System 2010 een grote sprong voorwaarts zal zijn op het gebied van Application Lifecycle Management.

Interesse?

Op 8 januari 2009, 19 februari 2009 en 16 maart 2009 organiseert Microsoft een VSTS 2010 Intrack.

Zie pagina 46 of http://www.msdnintracks.com/msdnintracks voor meer info

Marcel de Vries is Technology Manager bij Microsoft.