Arno Lubrun en Rossen Blagoev zijn beiden werkzaam bij Microsoft Services Nederland.

Visual Studio Team System voor projectmanagers

HOE OVERTUIG IK MIJN PROJECTMANAGER DAT HIJ TEAM SYSTEM NODIG HEEFT?

In de vorige twee nummers van .NET Magazine zijn twee artikelen verschenen die uitgebreid aandacht besteden aan de functionaliteit van Visual Studio Team System vanuit het oogpunt van de ontwikkelaar. Visual Studio Team System is echter een product dat niet alleen op ontwikkelaars is gericht, maar ook op de overige leden van een ontwikkelteam – de architecten, ontwerpers, testers, beheerders en projectmanagers. Met name voor de laatste groep is het vaak niet duidelijk hoe Visual Studio Team System hen kan helpen in hun werkzaamheden. Dit artikel bevat uitleg over wat projectmanagers hebben aan technische zaken als daily builds en FxCop, waar ontwikkelaars zo enthousiast over zijn. Tevens reikt dit artikel argumenten aan voor de ontwikkelaars om hun projectmanagers te overtuigen hoe zij ook kunnen profiteren van Visual Studio Team System.

De daily build is het kloppende hart van een project. Het daily build-proces is het proces waarbij elke dag de applicatie wordt opgebouwd als executeerbare applicatie op basis van alle code die er tot op dat moment gegenereerd is. Met de daily build ben je dus dagelijks precies op de hoogte. In het daily build-proces worden alle verschillende componenten van de applicatie gecompileerd en gecombineerd tot de uiteindelijke applicatie voor zover deze op dat moment klaar is. Daarna wordt de gehele applicatie getest op basis van de testscripts om te bepalen welke delen van de applicatie werkt. Dit lijkt een zwaar en bewerkelijk proces. Waarom niet aan het einde van het project een test doen en dan bepalen of alles werkt? Dat scheelt je immers dagelijks uren aan inchecken, testen, en dus uren!

De kneep zit hem in het volgende: in de praktijk zie je vaak dat aan het einde van de bouwfase de applicatie wordt opgebouwd, een test wordt uitgevoerd op de code en vele aanpassingen worden gedaan. De doorlooptijd en benodigde tijd voor de testfase (vaak integratietestfase) zijn niet nauwkeurig te plannen aangezien je niet genoeg inzicht hebt in hoe de verschillende onderdelen met elkaar samenwerken en welke issues hieruit voortvloeien. Pas als je de tests uitvoert, kom je erachter wat de applicatie allemaal nog mist. Hierdoor zul je in de praktijk vaak zien dat er nog veel onzekerheid is over de uiteindelijke opleverdatum en de planning, ook al is de applicatie reeds door de bouwfase heen. Daarnaast is de kans van 'fout op fout' groot. Er kan in het begin van de bouwfase een denkfout zijn gemaakt, die nu door de gehele applicatie wordt gebruikt en waarop vervolgens de overige code is gebaseerd. Kortom, grote herstelkosten.

De daily build zorgt ervoor dat je al begint met testen tijdens het bouwen. Hierdoor kun je in een vroeg stadium zien welke problemen er optreden. Het inschatten van de uiteindelijke benodigde doorlooptijd en uren voor het gehele project vindt in dit geval dan ook nauwkeuriger en in een vroeger stadium van het project plaats. Daarnaast is de kans van 'fout op fout' kleiner. Het inschatten van de benodigde tijd is ook mogelijk doordat je tijdens de bouwfase beter weet wat de stand van zaken is. Als je geen daily build hebt kun je als projectleider aan de ontwikkelaar natuurlijk vragen: "hoe ver ben je?", waarop het antwoord bij mij altijd is: "ik ben al een eind op weg" of "ik ben bijna klaar". Zo weet de projectmanager nog niets. De ontwikkelaar zit immers op een creatieve wolk en het laatste wat je hem dan moet vragen is een exacte inschatting van de tijd. De kans dat hij dit zelf weet is kleiner dan arbitrair gokken. Met daily build kan de projectleider aan de tester vragen: "hoeveel van de uiteindelijke functionaliteit werkt er in de daily build van vandaag?" Dit is wat mij betreft een veel exacter getal waarbij de voortgang veel beter inzichtelijk is. Kortom, wil je als projectleider niet verworden tot projectlijder, dan is een daily build wat ons betreft een must.

FxCop dwingt de .NET-codeerstandaarden af

In de praktijk zie je dat meer ontwikkelaars aan een applicatie werken, waarbij ieder zijn eigen stijl en eigen wensen heeft. Een grote valkuil die zeker in grotere applicaties optreedt, is dat de onderhoudbaarheid van de code tijdens het project minder wordt. De code van de ene ontwikkelaar is bijna niet leesbaar voor de andere en er worden overal andere codeerstandaarden gebruikt. Het gevaar van creatieve chaos is dichtbij. Nu heeft Microsoft dit probleem ook gezien en daarom zijn er ontwikkelstandaarden voor .NET gepubliceerd. De standaarden zijn regels waar iedere .NET-ontwikkelaar zich aan dient te houden. Prima zou je denken. Maar projecten zouden geen projecten zijn als onder druk van het creatieve proces toch concessies worden gedaan waar we later ontzettend spijt van hebben. Als voorbeeld nemen we het volgende ervaringsscenario: ja, gooi die code van die eerste versie maar weg en begin maar opnieuw. Codeerstandaarden? Daar hadden we geen tijd voor.

Afbeelding 1. Verschillende typen work-items

Een tool als FxCop biedt de mogelijkheid om eenvoudig de .NET-codeerstandaarden van Microsoft 'af te dwingen'. Wanneer niet alle FxCop-regels wenselijk zijn, of als extra regels gewenst zijn, dan is de tool wel aanpasbaar. Natuurlijk is dit niet genoeg om te garanderen dat absoluut alles op de juiste wijze wordt ontwikkeld, maar het kan wel helpen een eerste filtering aan te brengen. Zo kan de beheerorganisatie bijvoorbeeld eisen dat alleen applicaties geïnstalleerd mogen worden die zonder problemen door FXCop zijn gekomen. Dit is nu ook bij Microsoft een standaardeis voor alle .NET-code. Natuurlijk is FXCop, zoals alle tools, niet de ultieme oplossing: ook de cultuur en het werkproces zijn minstens zo belangrijk. Maar het helpt wel om software beter onderhoudbaar te maken.

Alle projectlijstjes worden 'work-items'

Een projectmanager moet verschillende overzichten bijhouden tijdens een softwareontwikkeltraject. Elk lijstje bevindt zich op een andere plek, in een andere tool. Voor de eisen (requirements, use cases) wordt vaak een combinatie van Excel en Word gebruikt. De taken voor het projectteam bevinden zich meestal in Microsoft Project op de laptop van de projectmanager. De bevindingen (bugs, defects) komen binnen via e-mail, of er wordt gebruik gemaakt van een zelfgebouwde Access-database. Naarmate het project groeit, groeit het aantal lijsten mee: de risicolijst, de lijst met issues, de change request-lijst, de besluitenlijst, en ga zo maar door. Zou het niet fijn zijn als al deze lijsten zich op één plek bevinden en op één manier worden behandeld? Dit is waar Work Item Tracking (WIT) in beeld komt. Elke lijst die de projectmanager moet bijhouden, kan worden bewaard in de repository van Visual Studio Team System in de vorm van een lijst van work-items. Met andere woorden, work-item is gewoon een verzamelnaam voor de inhoud van de lijst. Bevindingen, taken en change requests zijn allemaal verschillen typen work-items, zoals weergegeven in afbeelding 1.

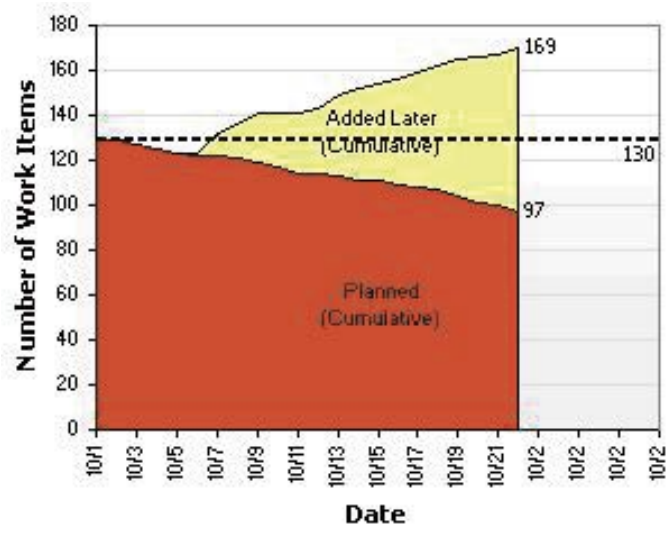
Afhankelijk van het type kan het work-item andere informatie bevatten. Bijvoorbeeld, voor een work-item van het type **bevinding** is informatie als probleemomschrijving, prioriteit, status, toegewezen aan en dergelijke van toepassing. Voor een work-item van het type **taak** zijn gegevens als geschatte uren, gebruikte uren en 'wanneer klaar' belangrijk. De typen work-items en de informatie in een work-item zijn aanpasbaar en kunnen (en moeten) worden afgestemd op de projectbehoefte. Als de projectmanager alleen taken, bevindingen en use cases moet bijhouden, maar bijvoorbeeld geen issues, dan werkt hij met alleen deze drie typen work-items en kan hij specificeren welke informatie elk work-item-type moet bevatten. Verder kunnen de work-items aan elkaar worden gerelateerd; bijvoorbeeld bevindingen aan use cases relateren, om inzicht te kunnen krijgen in de kwaliteit van iedere use case.

Elk work-item, onafhankelijk van zijn type, doorloopt een bepaalde levenscyclus, die ook bijgehouden moet worden. Een use case wordt bijvoorbeeld eerst ingediend, daarna beoordeeld, vervolgens afgewezen of ingepland, dan geïmplementeerd, daarna getest, enzovoort. De afspraken over de levenscyclus van de diverse work-items worden meestal vastgelegd op papier als onderdeel van het ontwikkelproces aan het begin van het project. Daarna worden ze vaak niet gevolgd als het project onder druk komt te staan, omdat dit veel discipline vereist. Dan zie je dat ontwikkelaars use cases of changes oppakken, terwijl ze nog niet officieel beoordeeld en geaccepteerd zijn. Of dat ontwikkelaars 'leuke dingen' bouwen, waar geen use case voor is. Dit is waar het woord 'tracking' (volgen) van Work Item Tracking aan de orde komt. De gemaakte afspraken over de levenscyclus kunnen worden vastgelegd in Visual Studio Team System. Ook kan Visual Studio Team System afdwingen dat elk work-item zijn voorgedefinieerde cyclus volgt. Dit - in combinatie met de rest van de functionaliteit van Visual Studio Team System - zorgt dat de afspraken ook daadwerkelijk gevolgd worden. Leuke dingen toevoegen aan het project, als deze niet direct gerelateerd zijn aan een use case of een change, zijn dan niet meer mogelijk. Aan de andere kant worden ontwikkelaars ook beter beschermd tegen vage use cases, omdat deze dan eerst de afgesproken ontwerpstap moeten passeren voordat ze klaar zijn voor implementatie.

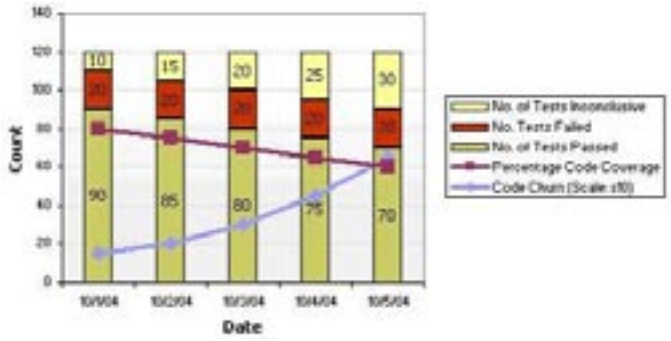
Bewerking van work-items in Excel of Project

Het heeft duidelijk zijn voordelen om alle projectlijsten op te nemen in de repository van Visual Studio Team System en op uniforme manier te behandelen. Maar het klinkt ook als de zoveelste nieuwe tool die een projectmanager moet leren beheersen. Het goede nieuws is dat work-items in de Visual Studio Team System-repository kunnen worden bewerkt met tools, waar een projectmanager al mee bekend is, namelijk Excel en Project. Feitelijk kan hij de lijsten blijven bijhouden in dezelfde tool en op dezelfde manier als tot nu toe. Alleen blijft de informatie niet in het Excel- of Project-bestand, maar gaat het naar de Visual Studio Team System-repository toe waar de informatie beschikbaar is voor iedereen en kan worden gerelateerd aan andere informatie zoals testresultaten, build-resultaten, enzovoort.

Visual Studio Team System is echter geen vervanger voor Microsoft Project. Project blijft de tool voor projectplanning. De integratie tussen de twee zorgt er alleen voor dat de planning in Project, als (work-items van type) taken voor de ontwikkelaars in Visual Studio Team System komt te staan. Deze taken kunnen verder gekoppeld worden aan andere Visual Studio Team System-gegevens. Planningsspecifieke functionaliteit zoals afhankelijkheden creëren tussen taken, kritische pad, gantt charts, baselines, enzovoort is geen onderdeel van Visual Studio Team System. Er is verder geen rechtstreekse koppeling met Microsoft Project Server gepland voor de eerste versie van



Afbeelding 2. Rapportage niet gepland werk: scope creep



Afbeelding 3. Correlatie van verschillende gegevens in de repository

Visual Studio Team System. Uitwisseling van gegevens is wel mogelijk via de Project-client.

De projectstatus is altijd actueel en inzichtelijk voor iedereen

Als alle projectlijsten als work-items opgenomen worden in Visual Studio Team System zijn ook uitgebreide rapportages en metrieken mogelijk. Dit aspect blijft vaak onderbelicht of beperkt zich tot het rapporteren van het aantal bestede uren versus het aantal use cases dat is opgeleverd. Het verzamelen van de benodigde gegevens en het maken van rapporten duren vaak dagen. Visual Studio Team System bevat standaard veel rapportages die inzicht in de projectstatus en de productkwaliteit bieden en die zijn gebaseerd op de informatie in de repository. Uitleg over wat de grafieken betekenen is ook inbegrepen.

Afbeelding 2 toont een voorbeeld van een rapport over niet gepland werk. Het project is begonnen met 130 work-items (use cases). Gaandeweg is een aantal use cases weggestreept en blijven er maar 97 over. Er zijn echter 72 nieuwe use cases bijgekomen en het totale werk is fors gestegen: van 130 use cases aan het begin tot 169 nu. Een duidelijk geval van erge 'scope creep'. Dit rapport zou - niet zonder moeite - ook zonder Visual Studio Team System kunnen worden geproduceerd. Waar Visual Studio Team System een stap verder gaat is met rapporten die correlaties aantonen tussen de verschillende gegevens in de repository, zoals het voorbeeld in afbeelding 3.

Dit rapport geeft een indicatie van de kwaliteit van het product op basis van de eerder in dit artikel toegelichte daily builds. Deze grafiek laat zien dat nog steeds nieuwe code wordt toegevoegd aan iedere daily build (code churn, blauwe lijn), maar zonder voldoende bijbehorende unit-tests (code coverage, paarse lijn) die de kwaliteit van de nieuwe code moet borgen. Sterker nog, het aantal mislukte bestaande unit-tests (de verticale balken) neemt toe. Met andere woorden, de kwaliteit van de bestaande code is achteruit gegaan en de kwaliteit van de nieuwe code is onbekend. Vanzelfsprekend kunnen de bestaande rapportages worden aangepast of nieuwe aangemaakt.

Het ontwikkelproces volgen

Work-items en rapportages zijn onderdeel van een groter geheel, dat ontwikkelproces heet. Of het nou MSF, RUP, XP of iets anders heet, het ontwikkelproces wordt meestal keurig vastgelegd aan het begin van het project, waarna het niet meer wordt gevolgd. De kernoorzaak is dat het volgen van het afgesproken ontwikkelproces discipline vereist. Als het project onvermijdelijk onder druk komt te staan, kiest iedereen voor de weg van de minste weerstand, omdat de gebruikte tools dat toestaan. Als het ontwikkelproces niet alleen op papier staat, maar feitelijk in de tools 'leeft', komen de gewenste weg en de weg van de minste weerstand overeen.

Visual Studio Team System biedt de mogelijkheid om niet alleen work-items en rapporten, maar ook andere aspecten van een ontwikkelproces in de repository vast te leggen. De belangrijkste van die aspecten zijn:

- De **rollen** binnen het ontwikkelproces (bijvoorbeeld projectmanager, ontwikkelaar, tester, enzovoort) en de **rechten** per rol binnen Visual Studio Team System. Hiermee wordt het bijvoorbeeld mogelijk om af te dwingen dat iedereen een wijzigingsverzoek mag indienen, maar alleen de projectmanager de wijziging mag autoriseren.
- De **iteraties** binnen het project (bijvoorbeeld 1, 2, 3), de **fases** per iteratie (bijvoorbeeld envisioning, planning, developing, stabilizing) en de op te **leveren** producten per fase en de **sjablonen** voor deze producten. Hiermee, in combinatie met de work-items, wordt het mogelijk om af te dwingen dat aan het eind van de planningsfase van iedere iteratie het testplan bijgewerkt wordt voordat het project verder mag.
- **Handleidingen (how to's)** die de diverse teamleden (projectmanager, ontwikkelaars, testers, enzovoort) helpen om hun werk te doen volgens het ontwikkelproces. Bijvoorbeeld, wat moet de analist doen om tot een goede use case te komen.

Op basis van al deze informatie dwingt Visual Studio Team System af dat iedereen die aan het project werkt zich conformeert aan het ontwikkelproces. Kortom, het wordt eenvoudiger om het ontwikkelproces te volgen dan om het niet te volgen.

Standaard ontwikkelprocessjablonen

Een team kan kiezen om hun huidige ontwikkelproces in Visual Studio Team System vast te leggen, maar het kan ook kiezen om te migreren naar een standaard ontwikkelproces. Een standaard ontwikkelproces uitkiezen en aanpassen is natuurlijk ook mogelijk. De voordelen van een dergelijke aanpak zijn analoog aan de voordelen van de officiële naamgevingconventie voor .NET. In het tijdperk van VB6 en C++ was een project vaak veel tijd kwijt aan religieuze oorlogen over hoofdletters, onderstrepen, accolades en dergelijke. Voor .NET heeft Microsoft een officiële naamgevingconventie gepubliceerd plus een bijbehorende tool die naamgeving controleert (FxCop). De projecten kunnen zich nu richten op het leveren van echte toegevoegde waarde. Dezelfde voordelen kunnen ook worden behaald indien men beslist dat het project een standaard ontwikkelproces volgt, waardoor projectteamleden hun energie ergens anders op kunnen richten.
Microsoft zelf gaat twee ontwikkelprocessen meeleveren met Visual Studio Team System. MSF for Agile Software Development is bedoeld voor kleinere teams en snelbewegende projecten. MSF for CMMI Process Improvement is bedoeld voor teams die op niveau 3 van het Capability Maturity Model (CMM) moeten of willen opereren. Er wordt verwacht dat andere partijen hun eigen ontwikkelprocessjablonen voor Visual Studio Team System op de markt brengen, zodat er genoeg keuze ontstaat voor een standaard ontwikkelproces en dat het beste bij de organisatie past.

Samenwerken

Visual Studio Team System ziet ontwikkelen van software niet meer alleen als proces voor ontwikkelaars. Het ondersteunt de trend dat softwareontwikkeling een teamproces is waarin diverse personen een rol en functie hebben. Visual Studio Team System kan daarom zeker de brug zijn tussen ontwikkelaars en projectleiders. Dit neemt niet weg dat het een tool is om het ontwikkelproces te faciliteren. Uiteindelijk gaat het erom hoe mensen samenwerken om de software tot een succes te maken.

Arno Lubrun en Rossen Blagoev zijn beiden werkzaam bij Microsoft Services Nederland. Arno is senior projectmanager en Rossen is Senior Developer Consultant. Rossen is 'Visual Studio Team System field champ' voor Microsoft Nederland.

Nuttige internetlinks

http://lab.msdn.microsoft.com/vs2005/teamsystem/
http://lab.msdn.microsoft.com/teamsystem/msf/