Thema

De RAD Race van Development Tools 2007 kende dit jaar vele teams met nieuwe technologische oplossingen. Maar liefst twee Ruby on Rails teams, en niet minder dan vijf tools die je 5GL's zou kunnen noemen, of open 4 GL's. In totaal deden aan de RAD Race van Development Tools 2007 elf teams mee. Gezien de technische verschillen tussen de teams en ook gezien hun scores leek het ons aardig ze hier allemaal aan bod te laten komen.

Development Tools RAD Race 2007

Veel nieuwe technologie

Het verschil tussen de score van het beste en het slechtste team bedraagt ongeveer een factor tien, maar daarmee haalt het minst goede team nog steeds een productiviteit die ver ligt boven die van een goed lopend project – terwijl we maar al te goed weten dat veel projecten helemaal niet goed lopen. Bovendien hebben sommige teams die dit jaar niet zo goed gescoord hebben in andere jaren gewonnen – kortom: ook de mindere goden verdienen aandacht, al komen ze ook in dit overzicht niet op de eerste plaats. De volgorde van de eerste zes teams is overigens ook die van hun score, bij de nadere teams is die willekeurig.

Jurering

Bij de jurering wordt gebruik gemaakt van juryscripts. Ieder jurylid beoordeelt een of twee modules, wat subjectiviteit van juryleden uitsluit. Het aantal punten per module staat van tevoren vast. Ontbreekt er iets wezenlijks aan de functionaliteit, dan krijgt het team 50% van de punten, voldoet de module op twee of meer punten niet, dan 0 %. Voor de GUI (gebruikersinterface) kunnen 0, 10 of 20% extra worden verdiend.

De jury bestond dit jaar uit:
* Rick van der Lans, voorzitter, R20 consultancy
* Ron Tolido, Capgemini
* Wouter Goedvriend, Capgemini
* Cor Baars, CIBIT
* Willem Koppenol, Twice IT
* Jos Warmer, Ordina
* Klaas Brant, KBCE
* Dré de Man, Array Publications

CrossmarX: Snelheid geen doel

CrossmarX won de RAD Race dit jaar net zoals vorig jaar. Het verschil was dat hun voorsprong op nummer twee nog groter was. In tegenstelling tot vorig jaar zag het team ook achteraf geen noodzaak om het tool aan te passen.

Een Java 4GL, zo noemden we de CrossmarX Application Engine vorig jaar. Zo zou je hem nog steeds kunnen karakteriseren, een 4GL-achtig tool dat uiteindelijk Java-code produceert die binnen Tomcat als servlets draait. Geen Beans, gewone pojo's in feite. Tien jaar geleden, toen CrossmarX begon met het ontwikkelen van het tool was het nog modern, tussentijds uit, en nu weer heel modern. De keuze voor deze standaardoplossing bekent ook dat het met de schaalbaarheid redelijk goed zal zitten, wat een aantal praktijkprojecten van CrossmarX ook bewezen heeft.
Het is dus geen 4GL in die zin dat het proprietary

Tekst en fotografie:
Dré de Man

code produceert. De manier waarop de code geproduceerd wordt, herinnert in zekere zin wel aan een 4GL, in ieder geval in die zin dat het een niet standaardmanier is, maar wel zeer gebruiksvriendelijk.
In feite maakt de CrossmarX Application Engine gebruik van een repository waarin een set Javaklassen zitten. Deze worden aangepast op basis van een aantal muisklikken, en op deze manier is een applicatie – zeker wanneer die gebaseerd is op database tabellen (hier MySql, maar alle JDBC-databases kunnen worden gebruikt) – zeer snel te configureren.
Er kan gebruik gemaakt worden van een ruime set aan standaardcomponenten, bijvoorbeeld voor role based security en logging. Daarnaast is het mogelijk om gebruik te maken van zelf te schrijven *plug ins*, die ook weer uit eenvoudige Javacode bestaan. Een aantal elementen van de CrossmarX Application Engine is gepatenteerd. In 1997 is de ontwikkeling van de Application Engine gestart, in samenwerking met universiteiten, partners en klanten. Het onderzoek dat ervoor is verricht, is financieel ondersteund door SenterNovem.
CrossmarX zet het product tot nu toe vooral in bij eigen projecten, maar een toenemend aantal klanten gebruikt het ook zelf. Het is expliciet de missie van CrossmarX om de tool (via het web) beschikbaar te stellen aan derden, partners, zelfstandige IT-bedrijven, IT-afdelingen, et cetera. Het is ook niet bedoeld voor ontwikkelaars, maar eerder voor business- en informatie-analisten en functioneel ontwerpers. De 4GL-functionaliteit – als ik het zo mag noemen – is ook door mensen zonder programmeer talent zeer goed te gebruiken, Moeilijker opgaven zijn op te lossen door zelf te schrijven plugins, of door plug-ins van anderen. Uiteraard kan CrossmarX ook plug-ins voor opdrachtgevers schrijven. Vele van de functionaliteit van de CrossmarX Application Engine is zo tot stand gekomen. CrossmarX denkt erover na om een aantal plug-ins op een open source-achtige manier voor en door gebruikers ter beschikking te stellen.
Er is handmatig getest op basis van handmatig geschreven testcases. De CrossmarX Application Engine bevat de mogelijkheid om automatisch testscripts uit te voeren, maar volgens het team loonde dat hier niet omdat er te weinig tijd was.
Het team heeft weinig voordeel gehad van het feit dat het van tevoren wist dat het een webwinkel moest bouwen: het had nog nooit een webwinkel gebouwd en ontbeerde ook de tijd om dat vlak voor de wedstrijd nog te doen. Er zijn – anders dan het deelnemen aan de RAD Race – op dit moment geen concrete marketing-plannen. Volgend jaar wil het team weer winnen, en dan van die overwinning gebruik maken om het tool goed in de markt te zetten – in een versie die nog nét iets gebruiksvriendelijker is.

Het team van CrossmarX: Michel Vogler (l) en Matthijs Lambooy

Het verschil tussen de score van het beste en het slechtste team bedraagt ongeveer een factor tien

Finalist IT, Ruby on Rails: Finalist IT herhaalde zichzelf met succes

Het had weinig gescheeld of dit team had niet meegedaan: te veel projecten. Pas op het oment dat er een weeny gepost was op Holland on rails meldde men zich aan. Toeval, of misschien ook wel een gevolg van het feit dat men het niet de gedacht dat een ander Ruby team zonder hen mee zou doen onverdraaglijk zou zijn.

Strikt genomen lijkt de toolset waarmee dit team ten strijde trok veel op die van een team dat ooit een slechte Java-prestatie leverde met een kale JDK en Notepad. Tetxmate, de texteditor van de Mac waarmee dit taam werkte verschilt niet zoveel van Notepad, behalve dat het er wat cooler uitziet en vooral klinkt.
Maar iets verder kijkend zijn de verschillen toch vrij groot: misschien nog wel het belangrijkste verschil zit in het framework Rails, dat Ruby veel extra snelheid meegeeft en dat veel taken van de gebruiker afneemt.
Ruby is echter ook een veel eenvoudiger en snellere taal dan Java, onder meer door dynamische typing. Ook de verdere syntax is heel eenvoudig, wat het schrijven van zeer compacte, eenvoudige

Het team van Finalist IT: Michiel de Mare en Remco van 't Veer

Er is handmatig getest op basis van handmatig geschreven testcases

en leesbare code mogelijk maakt. Ruby kent twee belangrijke principes: allereerst *Convention over configuration*. Conventie, dus standaardwaarden, hoeven niet vastgelegd te worden. Configuratie betreft dus alleen uitzonderingen. Een tweede belangrijke principe is DRY (don't repeat yourself): Dit principe pleit ervoor om zaken voor zover ze vastgelegd moeten worden maar een keer vast te leggen.
Ruby (zie ook het vorige nummer van Software Release Magazine, 2/2007) is ook bijna te zien als een DSL, (domain specific language) en is in ieder geval erg geschikt om er DSL's mee te schrijven. Door de ondersteuning binnen de JSE 6.0 van Ruby (zie alweer SRM 2/2007) is het zelfs mogelijk Ruby als specifieke DSL in combinatie met het Java-platform te gebruiken. In ieder geval is het op deze manier mogelijk snel veranderende business logica te schrijven. Aan de andere kant is het ook mogelijk om binnen Ruby Java- functionaliteit aan te roepen door gebruik te maken van de Ruby-syntax.
Alhoewel ROR nogal verschilt van de CrossmarX application engine, is er in het gebruik veel minder verschil. Het Rails framework brengt voor een belangrijk deel dezelfde functionaliteit mee als het CrossmarX tool. Door de eenvoud en de leesbaarheid van de code zou theoretisch zelfs deels dezelfde doelgroep gebruik kunnen maken: informatieanalisten en anderen zouden er goed mee overweg moeten kunnen.

De scores van dit team verschillen niet zo veel van die van CrossmarX. Het grootste verschil zit in de ontbrekende (extra) modules 15 t/m 18. Het team was dus net iets langzamer dan Crossmarx, maar uiteraard is net iets minder snel een betere uitdrukking.
Het maken van de opdracht viel het team mee vergeleken bij vorig jaar. Wel vond het dat het teveel tijd verdaan had met maken van eenvoudige schermen. Het expertsysteem en het maken van de admin-schermen kostten het team de meeste tijd. Er is getest is met zelf geschreven testscripts bij de meeste modules. De wetenschap dat een webwinkel gebouwd moest worden was voor hen geen bijzonder nuttige informatie – er is geen extra voorbereidingstijd gebruikt.
Volgend jaar wil het team een stabiel Ruby-framework gebruiken voor de admin-interface en nog meer test-driven werken. Ruby wordt bij Finalist nu, - in tegenstelling tot vorig jaar – voor eigen projecten ingezet. Hoewel ROR uiteraard geen eigen tool van Finalist is, vinden er wel marketing- inspanningen plaats: op 7 juni organiseert Finalist IT een Ruby On Rails dag in Amsterdam.

Holder: Ruby on Rails

Kleinst mogelijke afstand tot nummer twee

De uitslag van het tweede ROR-team verschilde slechts 0,2 % van het eerste, toch werd anders gescoord op de verschillende modules. Ook werd er – deels – van andere software gebruik gemaakt, namelijk van een zelf gebouwde administratieve plugin voor Ruby On Rails.

Het team vond de opdracht meevallen, al kostte de opdracht met de virtual salesman veel tijd,die zich niet echt vertaalde in punten. Bij het schrijven van business logica van module 12c (virtual salesman travel) ondervond het team moeilijkheden, naar eigen zeggen omdat er geen voorgedefinieerde functionaliteit aanwezig was om

Het team van Holder: Chiel Wester en Stephan Kaag

eenvoudig alle combinaties van array-elementen samen te stellen. Kennelijk had het team iets te weinig voorbereidingstijd. Dat het al wist dat er een webshop gemaakt moest worden, had maar weinig invloed, want er was te weinig voorbereidingstijd over om daar nog veel voordeel uit te halen. Er is handmatig getest. In vergelijking met het andere Ruby-team viel op dat het module 12c niet werkend kreeg. Dit team had echter weer wel module 17 af, en 11 b.

Volgend jaar wil het team vooral eerst kijken welke opdracht het meeste tijd gaat kosten en daar direct mee starten. Ondanks de korte voorbereidingstijd én het feit dat het team geen enkele ervaring had met de RAD Race, wist het een heel goede score neer te zetten: een zeer goede prestatie. Holder werkt vrijwel uitsluitend met ROR. Voor meer informatie over Ruby on Rails: zie de tekst bij het Team van Finalist IT.

OutSystems: net niet bij de eerste drie

Het team van OutSystems zette een opvallende prestatie neer. Het eindigde nét achter de nummers twee en drie, maar het verschil was zo klein dat het net zo goed andersom had kunnen zijn. OutSystems Platform is een platformonafhankelijke (J2EE of .NET) code-generator. Er wordt gebruik gemaakt van een visuele omgeving om alle lagen van een enterprise applicatie te produceren (user interfaces, database logica, business logica en security).

OutSystems zegt met hun platform ook de gehele life-cycle te ondersteunen, natuurlijk inclusief onderhoud maar ook die van applicaties, van het ontwikkelproces tot de verdere evolutie en onderhoud, en inclusief operations en support. Verder maakt het gebruik van de OutSystems agile methodology, die erg lijkt op SCRUM.

Het team van OutSystems: Francisco Menezes en Gonçalo Veiga

Of het te maken had met de nationaliteit van de ontwikkelaars weet ik niet, maar het was duidelijk dat het team wat moeilijkheden had met de tekst van de opgave. Uiteindelijk – na veel overleg - was het wel het enige team na CrossmarX en Finalist dat de virtual sales man opgave goed had, zij het maar voor de helft.
Het team zelf dacht dat het aan hun tekortschietende kennis van fotografie lag, maar juist op dat punt verschilde het niet van de meeste andere teams, ook niet van de winnaars. Waar de business requirements door het team begrepen waren, vonden zij het gemakkelijk om te implementeren. Prettig is dat OutSystems standaard (C# of Java) code produceert, die goed gedocumenteerd is. Deze code is ook onafhankelijk van de OutSystems software aan te passen. Omgekeerd is het ook vrij eenvoudig bestaande code in te passen. Er werden geen beperkingen van het tool ondervonden, met andere woorden: alles kon met de standaardfunctionaliteit opgelost worden. Het tool is bedoeld voor een grote groep, van ervaren ontwikkelaars tot business analisten met minimale technische vaardigheden.

Het team testte iedere module. Deels ging dat automatisch, aangezien het OutSystems Platform iedere iteratie op consistentie onderzoekt, via de zogenaamde *continuous integration approach*, die iedere release volledig valideert voor hem te publiceren. De business logica werd door het team apart getest en daarna geïntegreerd met de rest van het proces. Ook dit team kon niet echt profiteren van de wetenschap dat ze een webshop moesten maken, omdat er andere projecten waren die op hun wachten. Aan de andere kant was al veel van de benodigde functionaliteit in het tool zelf aanwezig.

Het team ziet wel degelijk ruimte voor verbeteringen. Niet in het tool, maar wel in de manier van werken. Het was zich aanvankelijk niet goed bewust van de consequenties van 100/50/0 regel bij de jurering (zie kader). Daardoor heeft

Het maken van de opdracht viel het team mee vergeleken bij vorig jaar

het team zich gericht op aspecten die later niet bleken bij te dragen aan de score. Volgend jaar wil het team beginnen het bestuderen van de puntentelling en jurering voor het programmeren (*overigens ook een advies dat in de opdracht was opgenomen, red.*). OutSystems gebruikt hun software (nu al op versie 4.0) voor alle projecten die het bedrijf uitvoert. Onlangs is de OutSystems Express Edition geïntroduceerd. Dat is een iets beperkte versie van het platform, dat tot vijf gebruikers geen licentiekosten kent.

Servoy: moeite met de virtual salesman

Servoy zou je ook een Java 4 GL kunnen noemen. Het gebuikt een set library's bovenop Java die het aantal te schrijven coderegels reduceert. Uiteindelijk wordt Java byte-code geproduceerd. Java en JavaScript kunnen gebruikt worden om bijvoorbeeld business logica te schrijven. In veel gevallen zal er gewerkt worden door op basis van databasetabellen schermen te maken en die aan te passen.

In principe is het mogelijk om de Java zonder de software van Servoy te onderhouden, maar dit is zeker niet eenvoudig. Het team scoorde dit jaar niet zo goed als vorig jaar. Het was van mening dat opgave 12c niet te maken was, een feit wat door een aantal teams gelogenstraft werd.

Het team van Servoy: Jan Aleman en Johan Compagner

Getest is bij iedere module. Na de eerste dag was het team van mening dat het vrijwel alles af had. Nochtans was het de tweede dag vrij lang bezig met testen en aanpassen van de applicatie, dus een zeker optimisme was dit team niet vreemd. Bij de opgaven met CRUD-functionaliteit scoorde Servoy goed (CRUD = create update delete, standaard data(base) bewerkingen). De virtual salesman opgave was echter niet goed uitgewerkt. Opvallend was dat het team zelf dacht dat er 'maar één if-je' fout stond, maar zelfs nadat ze de mogelijkheid kregen dat aan te passen, bleef de virtual salesman onbruikbare resultaten opleveren; met name de samenstelling van combinaties van objectieven liet te wensen over. Latere bestudering van de screenshots toonde ook aan dat het team bij de berekening van de diafragmascores de wortel trok uit een waarde in plaats van het kwadraat ervan te nemen. De jury kreeg de indruk dat de opgaven 11 en 12 het team nogal zenuwachtig hadden gemaakt, om niet geheel duidelijke redenen.

Al met al heeft het team het helemaal niet gek gedaan, maar het had wel duidelijk te lijden onder sterke concurrerende tools en teams. Misschien lag daarin ook de verklaring voor de nervositeit. Het tool is bedoeld voor zowel junior als senior ontwikkelaars. Het team is van mening dat het tool goed geschikt was voor de opgave. Servoy wordt momenteel door zo'n 600 ISV's en 120.000 eindklanten gebruikt. Door het Servoy ISV programma en een aantal campagnes hieromheen is het de bedoeling deze aantallen binnen 2 jaar te verdubbelen. Het licentiemodel van Servoy is gebaseerd op concurrent users.

GenWise met GenWise Studio

Goed begin

GenWise is voor de RAD Race in zekere zin een oude bekende: onder de naam Radventure deed een van de twee teamleden al mee met de RAD Race met de 4GL Clarion. Radventure won bovendien de RAD Race twee keer achter elkaar.

De scripttaal en de ervaring met de functionaliteit van de 4GL kwamen in GenWise goed van pas. Het is echter een moderne oplossing die C# genereert. Behalve het nieuwe tool GenWise Studio, werd Hibernate, NUnit, .NET framework/ASP.NET en Visual Studio 2005 gebruikt.

Daarmee combineerde het team eigen software met zowel commerciële als open source software, wat een moderne combinatie opleverde. Een voordeel van Genwise is de slimme combinatie van gegenereerde en geschreven code. Wanneer de applicatie veranderd wordt en er nieuwe code gegenereerd wordt, blijft de handgeschreven code staan. Bovendien is de handgeschreven code duidelijk te onderscheiden van de gegenereerde code. De opdracht op zich was voor het team zoals verwacht, aangezien er van tevoren was verteld dat het om een webshop ging. De vragen met betrekking tot de virtual sales man voor lenzen waren niet verwacht, maar vond het team wel interessant doordat het voornamelijk algoritmes waren.

Er was niet een specifiek onderdeel aan te wijzen dat het team de meeste tijd kostte. Hoe meer punten een bepaalde opdracht, meestal hoe meer tijd het kostte. Zo oordeelde men. Een eenmaal met de tool geproduceerde applicatie kan verder bewerkt worden met de Microsoft Visual Studio 2005 of met de gratis .NET SDK van Microsoft. Bovendien is het mogelijk om met externe tools gebouwde componenten en webpagina's te integreren in het GenWise Studio project. De applicatie maakt gebruik van bekende open-source projecten, dan wel het standaard .NET Framework. Het team heeft geen beperkingen ondervonden van het tool, wel mogelijke optimalisaties geconstateerd. Het tool is bedoeld voor zowel junior als senior .NET ontwikkelaars.
GenWise studio genereert automatisch Nunit tests om de connectiviteit en alle niet-destructieve acties (selects) testen op de database uit te voeren. Tevens hebben ze bij het testen van de 'shopping assistant'-algoritmes uitvoerig gebruik gemaakt van unit testing. Het team heeft de laatste uren van de vrijdag ook gebruikt om alles functioneel te testen. Het team vond het prettig om van tevoren te weten dat een webwinkel gebouwd moest worden. Van te voren werden daarom een aantal mogelijke scenario's doorlopen.

Het team van GenWise: Sebastian Talamoni en Ward Bekker

Volgend jaar zou het team eerder vragen naar extra opdrachten. Sommige waren namelijk redelijk triviaal te implementeren en vaak voor veel punten. Daar in zat óók het voornaamste verschil met Servoy. Genwise had over de gehele linie zelfs iets beter gescoord, in vergelijking met het andere team ontbraken alleen de punten voor twee modules van de extra opdrachten.

Genwise is een commercieel tool die wereldwijd verkocht wordt en tevens intern veel gebruikt wordt voor consultancy projecten. De marketing gebeurt nu vooral door de website en via een trial versie. Later moet dit meer aandacht krijgen, onder meer door online-advertenties.

iProfs: CRUD-functionaliteit liever gegenereerd

Het team van IProfs was het hoogst geklasseerde team dat zich niet van nieuwe technologie bediende: het schreef Java met Eclipse. Het gebruikte echter ook nog aantal tools, frameworks die het ontwikkelwerk versnelden: de plug-in oplossing MyEclipse, Spring framework, Hibernate, Adobe Flex Builder, Flex Data Services 2 en JRun als appserver.

Op zich zou met deze set een plaats bij de eerste vijf te halen moeten zijn. Maar al bij de eerste dag was aan het team te merken dat iets tegenzat. Hoewel zowel het team als de Adobe-software in andere projecten getoond hadden hun kwaliteiten te hebben, was het hier deze combinatie van team en tool die beide de das omdeed. Meer dan ooit geldt in de RAD Race: tools waarmee je niet kunt lezen en schrijven leveren binnen de korte duur van de Race geen winst op.
Het maken van de opdracht viel het team dan ook tegen, omdat het veel meer tijd nodig had voor het maken van de front-end van de applicatie dan

Het team van iProfs: Dennis van der Meer en Erik Jan de Wit

verwacht. Daarnaast kostte de virtual salesman veel tijd, terwijl het team daar geen punten mee scoorde. Het meeste last had het team naar eigen zeggen van het feit dat het een CRUD-applicatie niet heel snel kon genereren. Er moest dus heel veel zelf geschreven worden zonder dat daar delen van gegenereerd konden worden. Het team zag het zo: 'We konden gewoonweg niet op tegen de code- generatoren.' Toch had het team – behalve van de virtual salesman – heel wat functionaliteit werkende. Aan de ontbrekende delen was vooral te zien dat het allemaal net niet snel genoeg ging.

Voor het testen van de Virtual Salesman en voor wat DAO-objecten is JUnit gebruikt. Voor de overige code zijn geen tests geschreven. Het team had niet veel voordeel van het feit dat het wist dat er een webwinkel gebouwd moest worden, het maakte voor de door hen gekozen oplossingsrichting niet uit.
Volgend jaar zou het team in ieder geval een tool willen hebben die een standaard CRUD applicatie kan genereren. Het gaat tools voor Flex bestuderen, die het maken van een CRUD-applicatie zouden kunnen vergemakkelijken.

Het was de eerste keer dat het Flex heeft gebruikt. Hibernate en Spring worden voor alle projecten gebruikt. Het doel van het team was te onderzoeken of ze met behulp van Flex snel een Web 2.0-applicatie konden bouwen. Het antwoord daarop hebben ze nog niet gevonden, maar iets meer ervaring met Flex zal het wel brengen.

Avanade - Visual Studio 2005

Tijdverlies door onervarenheid met MCS 2007

Voor het tweede team dat met een klassieke omgeving ten strijde trok, gold vrijwel hetzelfde als voor het iProfs-team: onwennigheid met een deel van de toolset kostte teveel tijd. Het haalde ook een score die nauwelijks van die van het Java-team verschilde.

Aan de andere kant was al veel van de benodigde functionaliteit in het tool zelf aanwezig

Het team heeft gebruik gemaakt van de Microsoft Visual Studio 2005 IDE. Daarnaast is gebruik gemaakt van het Microsoft .NET Framework 2.0, Microsoft Commerce Server 2007 en Microsoft SQL Server 2005. De beslissing MCS 2007 in te zetten was een beetje geboren uit de paniek die ontstond toen het team hoorde dat het een webshop zou moeten gaan maken. Het team werkte meestal aan gecompliceerder opdrachten, custom development zoals het het zelf uitdrukt. Met het maken van een webshop had het eigenlijk geen ervaring en ook niet meteen een geschikt framework. MSC 2007 leek die functionaliteit te bezitten – en daarmee had het team eigenlijk wel goed gegokt. Alleen: doordat het MCS 2007 niet goed kende, verloor het heel veel tijd met het custom bouwen van functionaliteit die binnen MCS 2007 vrij eenvoudig te configureren was. Dat was jammer, want over de gehele linie scoorde het team niet slecht. Uiteindelijk wist het echter van de moeilijker onderdelen te weinig af. Net als bij iProfs was hier sprake van een interessante toolkeuze gecombineerd met een te korte inwerktijd. Het team verklaarde dat het plaatsen van de aangeleverde data verrassend genoeg veel tijd gekost heeft. Om dit goed in Commerce Server 2007 te krijgen, hebben ze een CS2007 webservice en een maatwerk import routine moeten gebruiken. Daarnaast heeft module 12 de meeste tijd ten opzichte van de andere modules gekost.
Aan Commerce Server 2007 of het Microsoft .NET Framework heeft het team geen beperkingen kunnen ondervinden. Met een gestructureerd applicatiemodel is het zeer eenvoudig om complexe businesslogica in een applicatie te implementeren. Het team heeft bij het afronden van iedere module handmatige tests uitgevoerd.

Volgend jaar wil het team meer aandacht richten op custom development. Commerce Server 2007 heeft veel out-of-the-box functionaliteit en voordelen met zich meegebracht, echter ook een tijdsverlies. Toch was het team niet ontevreden over het resultaat van de race in twee dagen. De webwinkel was klaar om 's avonds online geplaatst te worden. Dat klopte op zich, maar we hadden graag gezien wat het had gepresteerd wanneer het wat beter bekend was geweest met MCS 2007.

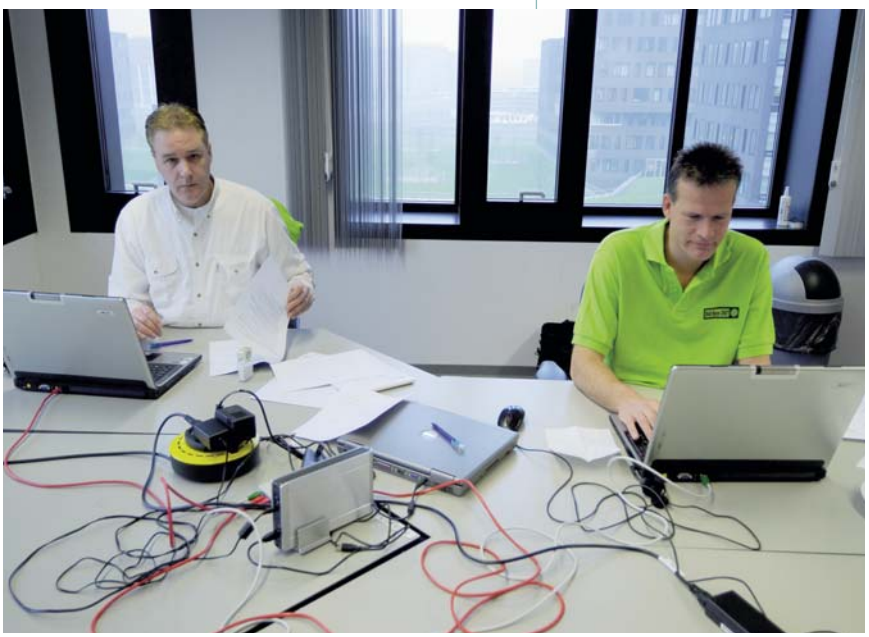
Magic Hands / Magic eDeveloper

Magic verloor iets van zijn magie

Magic heeft bij de RAD Race een naam hoog te houden. Niet alleen won het jaar twee jaar geleden de race nog, het heeft in de afgelopen jaren vaak een goede prestatie geleverd.

Over de gehele opdracht stelde Magic dit jaar een beetje teleur, tenminste wanneer je de prestatie zou meten aan die van eerdere jaren. Over de gehele applicatie was er overal wel werkende functionaliteit, maar er ontbrak ook overal wel wat. Overigens was Magic Hands ook het tweede team dat met hardwarepech te kampen had: twee dagen voor de race crashte een laptop. De mail waarin stond dat ze een webwinkel moesten bouwen werd daaddoor niet meer gelezen en daarmee was Magic minder goed voorbereid op de opdracht. Het werken met een Vista-laptop waarop Magic weer een virtuele XP-versie draaide zal ongetwijfeld het bouwen van de applicatie ook niet versneld hebben.
Het team gebruikte Magic eDeveloper, een Pervasive (OO-) Database en Adobe Dreamweaver, voor de verfraaiing van de webpagina's. Het team vond de opdrachten over de virtual salesman te complexe calculaties bevatten voor een RAD Race. De opdracht 12c kostte het team ook de meeste tijd, ook hier zonder punten op te leveren. Waarschijnlijk had het er beter aan gedaan die tijd in de andere modules te investeren. Het team gaf ook aan volgend jaar het laatste half uur te willen gebruiken voor een volledige eindtest. Nu waren er te veel modules die de indruk wekten met een klein beetje aandacht een veel betere score op te kunnen leveren.

Het team van Magic Hands: Peter Vos en E. Broekhuis



Unisys AB Suite:

Geen top-score, maar nog steeds een prestatie

De Unisys AB Suite is net als OutSystems in staat om zowel C# als Java te produceren. Daarnaast kan er ook COBOL gegenereerd worden, wat gezien de achtergrond van Unisys niet verwonderlijk is. Hij bestaat in feite uit een high level scripting taal, waarmee je op PIM level de specificaties van een applicatie en database vastlegt en van daaruit genereert naar of C# of Java of Cobol.

Wel kan de applicatie uitgebouwd worden met andere code, bijvoorbeeld C# of Java, en zijn er diverse interface mogelijkheden beschikbaar om de AB Suite applicatie op te nemen in een SOA architectuur.
De applicatie kan niet aangepast worden met standaard software maar dient op het PIM level middels AB Suite Developer aangepast te worden. Dit levert niet alleen een productiviteitswinst op, maar zorgt er ook voor dat het hardware- en software-onafhankelijk ontwikkelen ook in tact blijft. Voor het runnen van de applicatie is de AB Suite Runtime omgeving nodig. Dit is een framework van componenten die de nodige standaardfuncties leveren, zoals het bewaken van de integriteit van de database en bijvoorbeeld recovery functionaliteit. Het is bedoeld voor ontwikkelaars die niet houden van technische hoogstandjes, maar samen met de eindgebruikers business vraagstukken willen oplossen in applicaties die vele jaren probleemloos kunnen draaien en eenvoudig zijn aan te passen.
De wetenschap dat een webwinkel gebouwd moest worden leverde niet veel voordeel op

Het team van Unisys: Parthiban Jeyalakshmi Varadarajan en Mahesha Puttaswamy Gowda

omdat de tijd te kort was . Het is zelfs mogelijk dat Unisys niet mee gedaan zou hebben aan de race wanneer het voor de inschrijving geweten zou hebben dat het om een webwinkel zou gaan. In het algemeen heeft Unisys op alle onderdelen wel punten gescoord maar ook steekjes laten vallen. Het team had bijvoorbeeld van de moeilijke onderdelen ook wel iets goed, maar gaf toe dat het nog geen oplossing had gevonden voor het probleem van 12c.

Volgend jaar zou het team eerder vragen om wat voor soort applicatie het zou gaan. Ook zou het meer re-usable code en frameworks meenemen, sneller de requirements vertalen in code en sneller en eerder samenvoegen van de modules en het doen van integratie tests. Ook de GUI komt volgens het team voor verbetering in aanmerking. Dat laatste was ook de jury opgevallen, want dit team scoorde vrijwel geen extra punten voor de GUI. Deels had dat mogelijk ook te maken met culturele verschillen, deels lag dat aan de aard van het tool. Voor een tool als dit, dat vooral gericht is op grote en zeer schaalbare administratieve applicaties was de score toch niet slecht: het team heeft bewezen dat daarmee nog steeds veel malen sneller gebouwd kunnen worden dan in een goed werkend conventioneel project. Daarmee leverde het geen top-score, maar nog steeds een prestatie.

AB Suite is voornamelijk een tool dat Unisys verkoopt aan hun relaties om er zelf applicaties mee te bouwen. Unisys heeft er grootse marketingplannen mee, omdat het volgens Unisys op een hoog niveau applicatie-ontwikkeling mogelijk maakt en zowel naar .NET als JEE kan genereren. Unisys verwacht ook dat mede door de integratie met Visual Studio de acceptatie van AB Suite in de markt snel zal toenemen. Het is eenvoudig te leren en mede daarom zoekt het op dit moment HBO-instellingen die hiermee studenten willen laten werken.

Indicia: Murphy's law

Volgend jaar wil het team meer aandacht richten op custom development

Het team van Indicia zag zich dit jaar vooral geplaatst voor de uitdaging Murphy's law te ontkrachten. Daar is het niet in geslaagd. Het begon al voor het begin van de wedstrijd met een iPod waarop een library stond die er maar niet af wilde. Normaal werkt het team vanaf een server waarop alle nodige tools en frameworks staan, en het was er niet echt aan gewend deze mee te moeten nemen.

Ontbrekende library's bleven het team de eerste dag kwellen. Maar zoals dat hoort volgens Murphy, werd het de tweede dag nog erger: het team kwam op een zeer drukke snelweg zonder benzine te staan en werd uiteindelijk weggesleept. Al met al liep het team vele uren achterstand op en dat was ook aan de score te zien. Het team werkte wel met een toolset waarmee het in principe een redelijke prestatie had kunnen leveren: Eclipse 3.1, Struts 2, Spring 2 en Hibernate 3.
Het was ook van mening dat de opdracht op zich mee viel, maar miste (net als het andere Java-team, iProfs) de mogelijkheid om code te genereren, om tijd te winnen, standaardoplossingen in te zetten en fouten te voorkomen. We zijn in ieder geval benieuwd hoe het team met de mogelijkheid om delen van de applicatie te genereren zal scoren, maar denken dat het dan ook nog even moet kijken naar de mogelijkheid om met Murphy af te rekenen!