thema

Software Release Magazine liet met en op initiatief van Rick van der Lans een aantal leveranciers van tools die in staat zijn om vanuit UML Java-code te generen, een eenvoudige administratieve applicatie maken: *The Bates Motel*. Het leverde een verrassend overzicht van een verzameling nogal uiteenlopende tools met verschillende toepassingen. Sommige tools lijken ons een vooruitblik te geven op wat 5 GL tools zouden kunnen worden, andere leken meer modeling tools, alle bleken in meer of minder handige mate in staat Java-code te genereren.

Java coderen of genereren?

Met UML op weg naar 5GL

De douchescène uit Psycho is één van de meest bekende (moord-) scènes uit de filmgeschiedenis. De hele film draait om het *Bates Motel*, en zo heet de door Rick van der Lans gemaakte opgave voor de tools dan ook. Als er een overeenkomst bestaat tussen de film en de opgave, dan is het wel met de manier waarop de douchescène gemaakt is: voor 45 seconden film waren zeven filmdagen en zeventig cameraposities noodzakelijk.

In deze tijden waarin weer gekeken wordt naar duur en kosten van projecten, wordt de markt (weer) rijp voor productiviteitsverhogende tools. Dat was dan ook de idee achter deze bonte verzameling: allemaal zouden ze in staat moeten zijn vanuit UML code te genereren, en dat zou productief moeten kunnen zijn, zo was de gedachte.

De toolleveranciers moesten naar een opgave met een zelfs voorgegeven eenvoudig UML diagram simpele administratieve applicatie maken. Daarbij ging het meer om het onderzoeken van de mogelijkheden en beperkingen van het tool, dan om een zo compleet mogelijke applicatie die in zo kort mogelijke tijd gebouwd moest worden. Het was dus geen Rad Race (dit jaar overigens weer in nummer 8, zie ook www.radrace.nl).

In de applicatie zaten een aantal integriteitsregels, er moest op beperkte wijze gerefactored worden, er zat een test op de ondersteuning van inheritance in, er werd gekeken naar de mogelijkheid van push up, de ondersteuning van BMP en CMP en optimistic en pessimistic locking, en de mogelijkheid die het andere hand veranderen van databaseserver, app server en framework (met name EJB 1.1/2.0 en BC4J).

Alhoewel de applicatie vrij simpel was, werd op deze wijze toch vrij veel duidelijk van de mogelijkheden en onmogelijkheden van de tools. Samenvattend kan gezegd worden, dat er geen enkele tool was die vanuit UML direct alles kon genereren, al waren er wel die daar dichtbij kwamen.

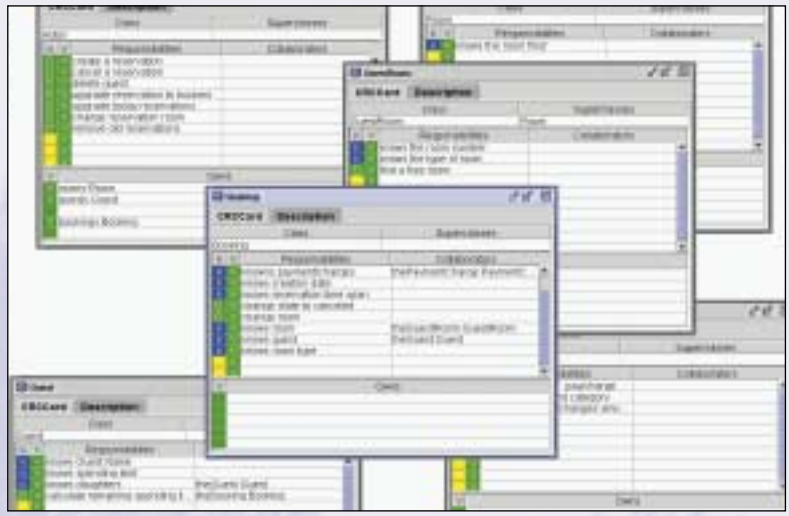
De belofte wordt dus nog niet helemaal waargemaakt. Maar de tools lijken wel de voorbode van een generatie, al is het nog moeilijk te zien welke van de tools de SQL-Windows of Powerbuilder van de toekomst zal zijn. Een korte bespreking per tool volgt hierna. Daarnaast zal Rick Van der Lans de tools uitvoerig bespreken op de door Array Publications – de uitgever van Software Release Magazine - georganiseerde avondconferentie van 3 december in de reeks www.expertmeetings.nl.

ArcStyler: MDA voor iedereen

ArcStyler neemt in deze reeks een bijzondere plaats in: het is het meeste MDA van alle tools. Het afwijkende van het tool zit hem vooral in de eerste stappen die men ermee zet: er wordt niet begonnen met een UML model, maar een zogenaamde BOM, een Business Object Model. Dit model komt tot stand door gebruikmaking van zogenaamde CRCcards.

De voordelen daarvan zijn - net als bij AdvantageGen - vrij duidelijk: de business logica is op vrij eenvoudige en platformonafhankelijke (PIM) manier in het programma aan te brengen. Overerving is overigens niet direct mogelijk. Deze methode van werken schijnt vooral ex-Cobol-programmeurs aan te spreken, al is het echter ook een uitstekende manier om met opdrachtgevers op een begrijpelijke, want niet in IT-termen uitgedrukte - manier over ontwerpen te spreken. Volgens de mensen van ArcStyler is het ook zaak een opdrachtgever het via de CRC-cards tot stand gekomen basisontwerp te laten tekenen. De uiteindelijke applicatie wijkt daar namelijk niet wezenlijk van af, en dus is onenigheid over het eindproduct terug te voeren op miscommunicatie over het BOM.

Ondanks het feit dat deze manier van werken een prominente plaats inneemt, is het niet noodzakelijk zo met de bouw van een applicatie te beginnen: er kan ook als bij de meeste andere tools vanuit UML gewerkt worden, en er kunnen zelfs UML modellen geïmporteerd worden. Daarbij treden dan natuurlijk de gebruikelijke beperkingen op: alle tools hebben zo hun eigen UML toevoegingen. Reverse engineering is ook mogelijk: import van code levert ook weer een model op. Het tool



kan niet alleen Java, maar bijvoorbeeld ook C# en zelfs Cobol genereren.

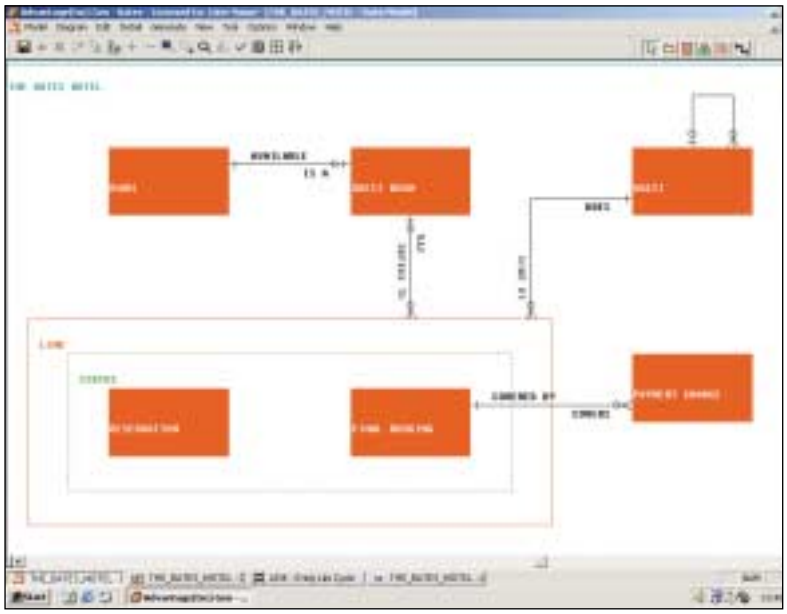
De uiteindelijke test ging vrij vlot, al moest er voor de een deel van de constraints wel echt geprogrammeerd worden. Overerving werd ook niet ondersteund, wat bij dit tool dus net als bij een aantal andere een consequentie is van de beperkingen van EJB 2.0.

Er kan met dit tool vrij snel code gegenereerd worden en de waarde ervan lijkt vooral in de architectuurgedreven manier van werken te zitten. Handgeschreven code wordt minder goed ondersteund, valt in ieder geval niet meer binnen het model terug te vinden. Voordeel is natuurlijk ook de platformonafhankelijkheid: aanpassingen binnen Java en zelfs overgang naar .Net projecten zijn vrij eenvoudig mogelijk.

Advantage Gen is Cool

Ondanks de hip aandoende vroegere naam is CoolGen (nu om CA-redenen Advantage Gen) binnen de groep van de hier onderzochte tools een oudje. Het bestaat al meer dan tien jaar. De nadelen van dit tool zijn duidelijk: het is geschreven in een tijd waarin UML nog niet eens bestond terwijl ook achteraf geen UML-ondersteuning is aangebracht. De voordelen ook: het werkt in sterke mate volgens een PIM en er kan op eenvoudige wijze zeer snel mee gewerkt worden.

AdvantageGen is niet alleen ouder dan UML, het is ook ouder dan Java. Maar omdat het bijna maximaal volgens PIM (platform independent model) werkt, was die ondersteuning er echter vrij gemakkelijk in aan te brengen. Gen kan ook C-code generen. In feite wordt pas na export naar het zusterprogramma AdvantageJoe een platform of taal vastgelegd. Zolang binnen de eigen taal (Gen, eigenlijk C met wat toevoegingen) wordt gewerkt is heel veel mogelijk. Stukjes code kunnen binnen het model hergebruikt worden, waarbij automatisch aanpassingen uitgevoerd worden (copy substitute geheten). De business logica en consistency checks kunnen uitgevoerd worden met behulp van voorgegeven



Entity Relation Diagram, weliswaar geen UML, maar wel zeer bruikbaar

constructies, zodat in feite geen kennis van een programmeertaal noodzakelijk is. 'If then else' constructies kunnen als het ware bij elkaar geklikt worden, wat voor menige Cobol programmeur waarschijnlijk een opluchting is. De ervaring met C-programmeurs die overstappen op Gen leert zelfs dat kennis van C in zekere zin een nadeel is: C-programmeurs hebben soms de neiging in de code zelf te gaan hacken, wat niet-gedocumenteerde code oplevert die bovendien niet meer herbruikbaar is.

Business logica en restricties zijn zo vrij gemakkelijk aan te brengen. In vergelijking met sommige andere tools, zou er wat meer wizards aangeboden kunnen worden. Aan de andere kant blijkt het werken met de voorgegeven booleans et cetera toch vrij snel, en het is ook een manier van werken die lijkt op het pure Java-code schrijven wat in vele andere tools noodzakelijk is voor dit soort opgaven, met het voordeel van het feit dat je binnen de beschermde omgeving van Gen blijft: code is herbruikbaar, verplaatsbaar en min of meer gedocumenteerd. Het zo platform onafhankelijk mogelijk werken heeft natuurlijk ook nadelen. Het aanpassen aan het platform gaat in principe buiten de ontwikkelaar om: het implementeren van andere frameworks staat wel heel ver van de Gen-manier van werken. Bovendien zijn aanpassingen van de code vanuit het model moeilijk, zo niet onmogelijk maken. Een ander nadeel: in zekere zin is een vorm van primitieve inheritance mogelijk, maar in feite wordt het niet ondersteund.

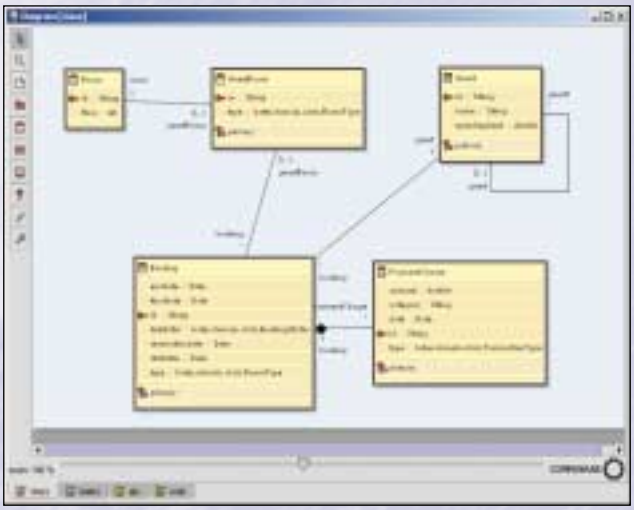
Positief is weer dat er een verzameling van third party templates bestaat, waarmee in feite onze opgave binnen zeer korte tijd gemaakt had kunnen worden. In de praktijk blijkt dat bedrijven vaak werken met eigen templates.

De opgave zelf bleek vrij snel met CoolGen te maken en leverde weinig problemen op voor de overigens wel zeer handige programmeur. Hij bleek pas 27 te zijn en had al vijf jaar Gen-ervaring. Dat lijkt een beetje op de droom van menige werkgever, maar de verklaring ligt in het feit, dat Gen aan de TH Twente gedoceerd wordt. In feite komt Advantage Gen het dichtst in de buurt van een 4GL. Ondanks de leeftijd kun je er moeilijk omheen toch te constateren: Advantage Gen is cool.

OptimalJ: optimaal gebruik van patterns

OptimalJ werd ons gedemonstreerd door de mensen die het geschreven hebben. Dat voerde tot heel veel uitleg over hoe het tool ontwikkeld is, en over de manier waarop patterns geïmplementeerd worden. Het beeld van OptimalJ was vooral dat van een tool dat op zeer consequente wijze in elkaar gezet was, al blijft er nog heel wat werk over voor volgende versies.

De essentie van het verhaal van de ontwikkelaars (daarover meer in het volgende nummer van Software Release Magazine, waarin een gesprek met chief architect Wim Bast zal zijn opgenomen) komt er kort gezegd op neer dat OptimalJ gebruik maakt van een multilevel model waarbij code patterns gebruikt worden en gebruik gemaakt wordt van transformaties van de ene laag naar de andere, in feite ook weer patterns. Op dit moment ontbreken er nog wat patterns, zodat de beperkingen van het tool overeenkomen met die van veel concurrenten: geen inheritance, geen joins op databases, geen pull up push down, maar wel optimistic locking. Het is echter redelijk eenvoudig om andere patterns te implementeren, hetgeen een van de grote voordelen van OptimalJ vormt. Dat neemt niet weg dat het tool echt Java-kennis vereist, code generatie voor gevorderden dus. Ook overeenkomend met veel concurrenten: 'handgeschreven' code gaat niet automatisch mee



met aanpassingen van het model, daarvoor wordt gebruik gemaakt van een zogenaamde dirty flag.

De start van de bouw van de applicatie gebeurde vanuit een klasse diagram. Het kan natuurlijk ook anders, bijvoorbeeld via een dbms diagram, of via reverse engineering. Aardig is dat constraints vrij gemakkelijk aan te geven zijn. Voor verdere business logica is wel al snel echte Java-kennis vereist. Reverse engineering is mogelijk, met name vanuit database tabellen. UML ex- en import via XMI kent net als bij andere tools zijn beperkingen door gebruikmaking van verschillende extensies.

Dankzij de vaardigheid van de ontwikkelaars was onze applicatie snel klaar. Ook de code generatie ging zeer snel in zijn werk, een van de sterke punten van OptimalJ.

Het beeld dat overblijft is dat van een tool waarmee nu al snel gewerkt kan worden, mits men over nogal wat Java-kennis beschikt. In de toekomst zou OptimalJ, wanneer het model consequent wordt uitgebouwd, kunnen uitgroeien tot een zeer krachtige en flexibele code genererende tool. Het is echter ook nu al een interessante tool, met de eerder genoemde beperkingen en voordelen.

JBuilder: zeer goed IDE

JBuilder is al enige tijd marktleider bij Java-IDE's. In een bundel met Rational Rose is het ook mogelijk om Java-code te genereren. Als echte codegenerator kan het niet gezien worden, maar in samenwerking met de zeer krachtige IDE biedt het wel mogelijkheden, zeker voor de toekomst.

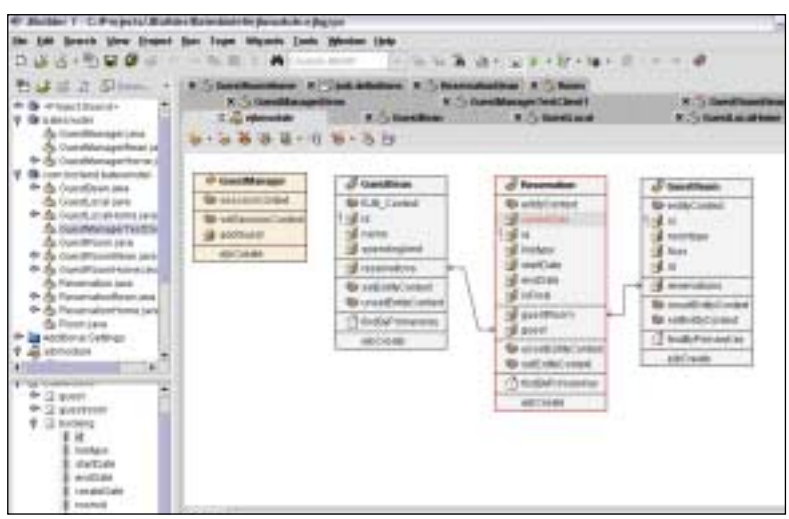
Van de reeks hier besproken tools is JBuilder het meest duidelijk in zijn aspiraties: het is in de allereerste plaats een IDE. Dat blijkt ook al uit het feit dat Borland een demoversie van ArcStyler meelevert. In de Enterprise Studio Edition levert Borland een zeer compleet pakket: Rational Rose, Borland Enterprise server, Macromedia Ultradev MX, Optimizeit, JDatastore, en ik zal ongetwijfeld nog wat vergeten zijn, zelden zal een software pakket in een dikkere doos verpakt zijn.

Een sterk punt van JBuilder is de ondersteuning van inheritance. Ook reverse engineering en refactoring werken heel goed, veranderingen in de code en het diagram blijven bovendien synchroon. Daarnaast zijn er natuurlijk de van Borland bekende wizards, die ook zonder het genereren van code in de zin van deze test, al veel productiviteitswinst opleveren.

Op het gebied van de instellingen ten behoeve van de aanpassing aan het platform (bijvoorbeeld EJB's, zie afbeelding) biedt JBuilder veel comfort. Verder heeft de ontwikkelaar door JBuilder veel controle over de applicatie, performance en scalability zijn zo goed in de hand te houden. Gebruikmakend van XMI is het mogelijk naar Delphi te exporteren.

Het maken van de applicatie ging snel, al bleef de vraag of het vanuit de BeanDesigner niet net zo snel of sneller zou zijn gegaan.

JBuilder blijft dus wat het altijd al was: een heel goed IDE. Het is mogelijk om vanuit UML code te genereren, maar menig ontwikkelaar zal het toch gewoon vanuit



De EJB Designer

Adv.
BEA

de BeanDesigner doen. In de praktijk zal het natuurlijk ook sterk afhankelijk van de functiescheiding binnen projecten zijn hoe het werkelijk gebeurt. Op navraag bleek dat Borland's Enterprise Studio for Windows veel beter in staat was om zonder veel code te schrijven snel applicaties af te leveren. Het zou wel heel vreemd zijn, wanneer JBuilder over een tijdje het kunstje niet van zijn Windows-broertje heeft afgekeken. Borland ziet MDA wel als een speerpunt, wat wel heel nieuwsgierig maakt naar de volgende versie van JBuilder.

Oracle JDeveloper: alles naar Java

"Lang leve BC4J"

Enkele jaren geleden besloot Oracle om de gebaande paden te vertalen en een grote schare PL/SQL-ontwikkelaars het leven iets moeilijker te maken: Oracle ging helemaal over naar Java. Ook de eigen applicaties werden in Java geschreven. Ook het oorspronkelijk van Borland gekochte JDeveloper werd helemaal herschreven naar Java. Maar JDeveloper is meer dan een herschreven JBuilder; dankzij de Oracle frameworks kan er snel mee gewerkt worden, met of zonder UML.

Oracle's stap om zich bij het Java-kamp te voegen zal waarschijnlijk meer ingegeven zijn door anti-Microsoft, dan door technische motieven. Voor Java en voor Oracle betekent het ook een enorme verandering. Op dit moment zijn er nog niet zo heel veel ontwikkelaars die de stap al gemaakt hebben. Om dat te vergemakkelijken heeft Oracle ook JHeadstart ingevoerd (zie ook: Optimize nummer 6, 2001).

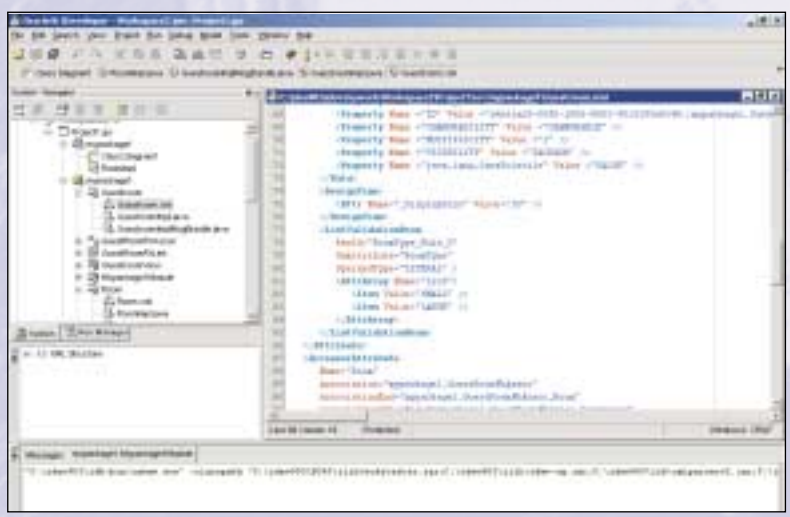
JDeveloper maak het mogelijk om met een UML-diagram te beginnen. Daarbij ook een activiteiten diagram dat eigenlijk bedoeld is voor Oracle's Integratie hulpje OAI. Met behulp daarvan is het ook mogelijk code te genereren, al zal dat eerder een skeletachtige vorm hebben.

In de praktijk is het misschien handiger om vanuit de Business Components te starten. Deze laatste maken het ook mogelijk om eenvoudige business logica te implementeren, waarbij redelijk eenvoudig constraints zeer eenvoudig aangegeven kunnen worden. Opmerkelijk binnen dit gezelschap, is dat inheritance ook ondersteund wordt: hierbij wordt ook gebruik gemaakt van BC4J. Push up pull down is ook mogelijk. Daarnaast kan er ook omgekeerd gewerkt worden: vanuit een bestaande database structuur waarbij wanneer met JHeadstart is gewerkt zelfs triggers en procedures meegenomen kunnen worden.

Zowel SQL als Java code kan gedebugged en geanalyseerd worden. Verder is er zowel een XML als een HTML editor geïntegreerd, die automatisch XML- en HTML-fouten aangegeven.

In de praktijk bleek dat de Oracle consultants vliegensvlug een werkende applicatie hadden opgeleverd. De voorgegeven constraints werkten, iets moeilijkere constraints niet. Uiteraard werd het eigen framework ondersteund, en was het ook mogelijk te kiezen voor optimistic locking. Voor de business logica moet wel vrij snel gewoon Java geschreven worden, al kan in voorkomende gevallen JHeadstart een tussenoplossing zijn.

JDeveloper lijkt het bewijs te zijn van het feit, dat Oracle het zeer serieus meent met Java. Voor Java is dat - anders dan sommigen denken - ook misschien wel goed: het framework maakt het mogelijk veel sneller te werken dan met pure Java, en zou ook richtinggevend kunnen zijn voor anderen. Ondanks dat alles blijft het heel wel mogelijk met JDeveloper slechte applicaties te bouwen. Een fool met een tool blijft een fool, en zelfs iemand die geen fool, maar evenmin een goede Java-programmeur is, kan al veel onheil aanrichten. Een aantal van de met JDeveloper geschreven applicaties zal dan ook tekortkomingen vertonen, met name op performance gebied. Maar dat valt JDeveloper nauwelijk te verwijten.

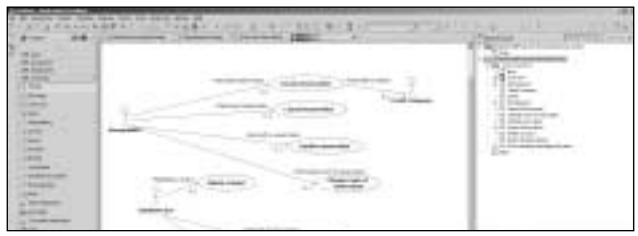


Rational XDE

Hulp bij de grotere projecten

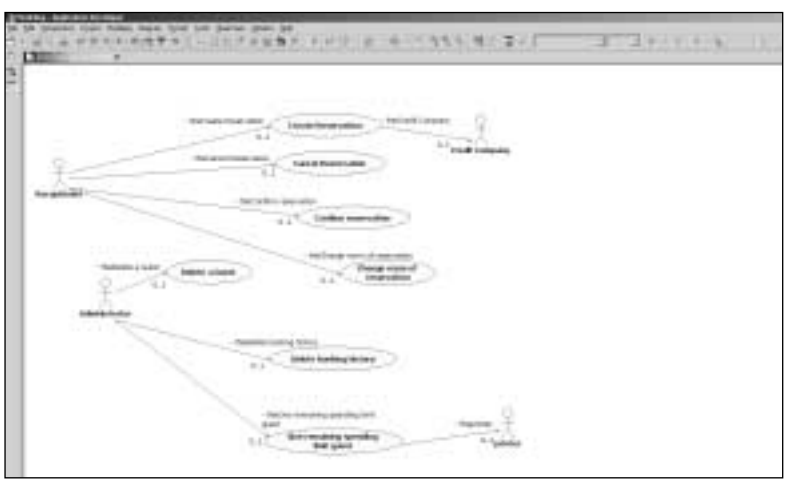
Wie Rational denkt, denkt aan UML. Geen wonder dat je verwacht dat Rational een tool biedt dat het genereren van Java-code vanuit UML mogelijk maakt. Dat blijkt inderdaad het geval, al moet bij het tool eerder aan UML, dan aan code genereren worden gedacht.

XDE betekent niets minder dan Extended Development Experience. De tool is niet alleen in staat Java code te genereren, maar ook C++, VB.Net. ASP.Net en C#. In latere versies zal daar nog onder meer Ada (!) en Pearl bijkomen. De ondersteuning voor diverse talen, laat ook een van de voordelen van XDE zien: het is mogelijk om een project van Java naar C# te 'vertalen', al zal dat – afhankelijk van de hoeveelheid taalspecifieke code – niet altijd even eenvoudig in zijn werk gaan. Bij grote projecten – denk aan fusies en dergelijke – zal zoiets toch wel eens voorkomen en dan is het een aardige bijkomstigheid. Verder moet dit tool vooral gezien worden als een hulpje bij grotere projecten die dan natuurlijk bij voorkeur met RUP worden uitgevoerd. Het werken volgens een architectuur aan het vasthouden daaraan zal er zeker mee vereenvoudigd worden. Wie een snelle Java-code generator voor een klein project zoekt, zal eerder voor een van de andere tools uit dit artikel kiezen. De Java-versie van XDE is geïntegreerd in WSAD (zie ook Software Release Magazine nummer 2, april 2002). Sommige van de eigenschappen van WSAD worden dan ook in XDE ondersteund, sommige niet. Omgekeerd zijn er weer dingen die bij XDE in sommige andere talen wel, maar bij Java niet worden ondersteund. Het hergebruik van code wordt door WSAD beduidend beter ondersteund dan door XDE. Zomaar overgaan op IBM's oplossing daarvoor gaat ook niet,



want het brengt modelinconsistentie met zich mee. Een latere versie van XDE zou dit echter wel moeten kunnen. Reverse engineering daarentegen weer wel, zelfs op basis van 'ruwe' code is het mogelijk een UML model te ontwikkelen en verder te werken.

Zoals eerder opgemerkt lijkt XDE in deze versie vooral geschikt om te werken in een omgeving waarin strak de hand gehouden moet worden aan een bepaalde architectuur. Volgens Rational kan het vasthouden aan de architectuur wel driehonderd procent productiviteitswinst opleveren. Die claim is wel gebaseerd op onderzoek. Natuurlijk zijn er ook genoeg situaties denkbaar waarin heel andere cijfers te halen zijn. Wie in een heel klein eenmalig project werkt, zal zeker zijn heil elders zoeken. De zogenaamde *reusable assets* zullen een op dit moment moeilijk in te schatten tijdwinst opleveren. Deze zijn voor Rational gebruikers gratis beschikbaar en zijn in feite codebrokstukken die soms goed bruikbare complete applicaties inhouden. Maar er zullen ongetwijfeld projecten zijn, waarin met XDE productiviteitsverhoging bereikt zal worden. Bij grote projecten speelt ook een andere eigenaardigheid van XDE zijn troeven uit: al is het genereren van code beperkt, het is weer wel mogelijk om zoiets zelf te automatiseren. Het is overigens wel jammer dat OCL nog niet ondersteund wordt.



Sygel's Wonder Machine Enterprise Edition

Een wonderbaarlijk tool

Sygel is een klein Belgisch bedrijf, dat erin slaagde na de dotcom crash tegen de stroom in toch te groeien. Hun product is misschien wel het meest duidelijke voorbeeld van een tool dat uit UML Java-code genereert: het doet precies wat je verwacht, maar alle voor- en nadelen vandien. Het is een eenvoudige en snelle codegenerator met wat nadelen, die misschien wel inherent zijn aan de gemaakte keuze.

The Wonder Machine, noemt Sygel hun tool, en hun strijdkreet luidt: draw it and run it! Dat klopt ook redelijk; vrij eenvoudige rechttoe rechtaan applicaties laten zich inderdaad op verbluffende eenvoudige wijze maken op basis van UML, en dat gaat ook enorm snel. De WMEE (Wonder Machine Enterprise Edition) is verkrijgbaar als plug in voor Rational Rose, Together ControlCenter en ArgoUML, een open source UML case tool, dat aan de universiteit van Californië ontwikkeld is. Het ondersteunt IBM WebSphere Application Server, SilverStream Application Server, BEA WebLogic Server, JBoss en Tomcat, ATG Dya en Jonas Open Application Server. Als databases worden ondersteund door Oracle 8i, MS SQLServer, DB2, Sybase SQL Anywhere, MySQL, Lutris InstantDB en HyperSonic database.

De werking van het tool is precies zoals je je die voorstelt. Je begint met een of meer UML diagrammen en geeft dan opdracht Java-code te genereren. In de praktijk blijken de Sygel mannen dan ook in staat in verbluffend korte tijd de hun opgedragen applicatie te kunnen bouwen. Constraints moeten echter keihard in Java-code geprogrammeerd worden. Andere beperkingen blijken ook: inheritance is niet mogelijk volgens J2EE 2.0 en dus ook niet met Sygel's WMEE. Wel denkt men bij Sygel erover na om reeds bestaande patterns te integreren, afhankelijk van de vraag hoe lang het gaat duren voordat inheritance in EJB's opduikt. Ook het gebruik van andere frameworks, zoals Oracle's BC4J, is niet mogelijk zonder het 'erven' van veel code, waarbij het voordeel van de code generatie alweer gedeeltelijk wegvalt. Reverse engineering is mogelijk op basis van database tabellen en UML, maar niet op basis van code. Ook zelf geschreven code komt niet meer terug in een UML diagram. Ten slotte is de UML deels wel proprietary.

De Wonder Machine doet heel snel wat je van een code generator zou verwachten, en dat is op de keper beschouwd toch bijzonder. Wanneer van de gebaande paden afgeweken moet worden, bij het implementeren van andere patterns bijvoorbeeld, werkt het natuurlijk ineens allemaal veel minder snel. Maar dat lijkt ook min of meer inherent aan de gekozen weg. Als pure code generator werkt de Wonder Machine echter wonderwel.

De WMEE wordt gemaakt door een klein team, ongeveer even groot als het groepje kantinedames dat de Compuware OptimalJ-ontwikkelaars bijstaat. Het blijken echter wel zeer goede programmeurs (de heren dus) en ze zullen zeker in staat zijn Sygel's WMEE nog aanzienlijk te verbeteren. Menige grote concurrent zou zo'n wonder machine en haar geestelijke vaders maar al te goed kunnen gebruiken. Desgevraagd blijken de mannen bij Sygel vastbesloten te zijn om zelfstandig verder te gaan, ook gezien hun ervaringen met venture capitalists bij eerdere werkgevers. Een wonderbaarlijk tool gemaakt door een dito team.

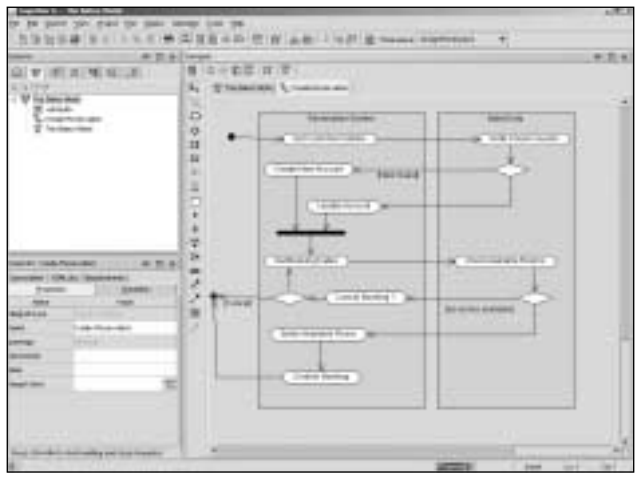
Togethersoft Control Center: samen sterk

Modelleren en genereren samen in een tool

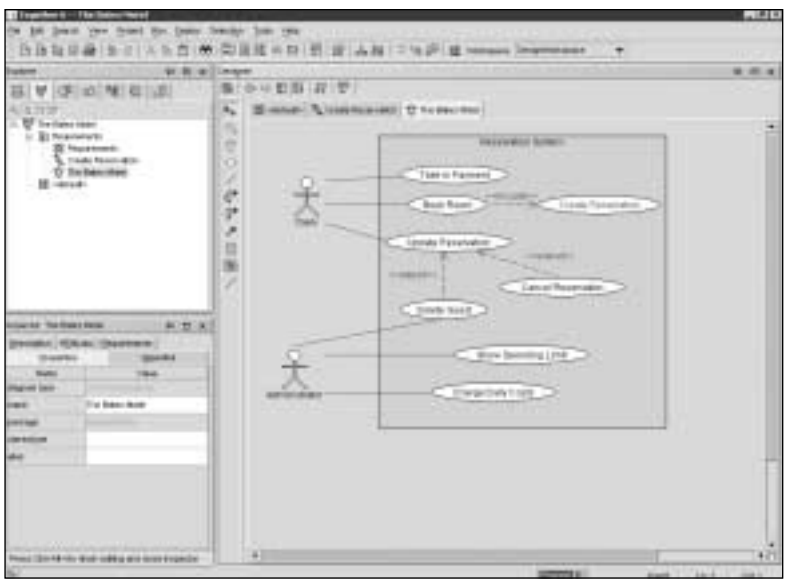
TogetherSoft is ontstaan uit een DAF, een Deutsch-Amerikanische Freundschaft. Twee ontwikkelaars brachten ieder hun eigen applicatie en ideeën in. Hoewel Togethersoft het een model build deploy platform noemt, zou het ook een model/build tool genoemd kunnen worden, want zo is het ontstaan.

Alhoewel officieel pas ontstaan in 1999, is de basis van het tool dus al veel ouder. De DAF bestond al sinds 1994, resulterend in TogetherJ en TogetherC++. Modeling, het genereren van code en deployen is functioneel goed gescheiden, wat ook resulteert in het feit dat behalve Java-code, ook C++, C# en VB.Net code gegenereerd kan worden. Het ondersteunt alle UML modellen en er is een goed IDE, geboren uit TogetherJ. Bij het genereren van andere code dan Java, is de IDE wel veel minder comfortabel. Door de flexibele opbouw is er veel integratie met andere producten mogelijk, bijvoorbeeld met Telelogic Doors, Starbase Caliber. Een Sun Java2 SDK versie 1.3 is al gebundeld met het product. Er is ook een uitgebreide testunit geïntegreerd. Alle JDBC-compliant databases worden ondersteund en alle bekende applicatieservers.

Inheritance wordt bij Together ondersteund. Gezien het gebrek aan ondersteuning binnen J2EE is dat niet vanzelfsprekend (zie ook elders in dit artikel). Prettig zijn de uitgebreide refactoring-mogelijkheden, waarbij ook push up en pull down goed werkt, al wordt het hier push down en pull up genoemd. Het implementeren van business logica kan voor een goed deel gebeuren vanuit het sequence diagram, constraints zullen echter vrij snel gewoon gecodeerd moeten worden. Aan de andere kant zijn code templates en snippets beschikbaar, die zeer gemakkelijk in een applicatie geïntegreerd kunnen worden. Er zijn vrij uitgebreide geïntegreerde code-analyse- en testmogelijkheden, XML en webservices worden ondersteund, pessimistic locking niet.



Een activity diagram



Use case demo

Het bouwen van de applicatie ging heel goed, waarbij met name de uitgebreide mogelijkheiden voor modellering opvielen. Toch moet Togethersoft Control Center vooral niet uitsluitend als een modeling tool gezien worden. Daarvoor werkt de codegeneratie en IDE te goed. Wie het negatief zou willen formuleren, zou kunnen zeggen dat het product op twee gedachten hinkt. Er kan echter evengoed gezegd worden dat bij Togethersoft de kwaliteiten van beide samen komen en in die zin is het op zijn beurt ook weer een uniek product.

Dré de Man