Tijdens de JavaOne-conferentie – medio vorig jaar – was het er ineens: OptimalJ van Compuware. Het gaat hier om een ontwikkelomgeving voor bedrijfskritische applicaties gebaseerd op Java 2 Enterprise Edition, waarbij de complexiteit van het objectgeoriënteerd werken wordt afgeschermd van de ontwikkelaar. Voordeel: ook Java-programmeurs met een minder diepgaande kennis van J2EE en objectgeoriënteerd werken moeten nu in staat zijn volwassen Java-applicaties te bouwen. Bovendien bestaat er nu een duidelijke koppeling tussen specificaties en uiteindelijke code. Interessant detail: OptimalJ is ontwikkeld in Amsterdam.

bespreking

OptimalJ: J2EE zonder diepgaande Java-kennis

Modelmatig werken in IDE-omgeving

Op diverse plaatsen wordt de discussie inmiddels al gevoerd: is er met recent geïntroduceerde producten als Describe van Embarcadero of Compuware's OptimalJ nu sprake van een wederopstanding van het fenomeen 'integrated development environment' (IDE) of gaat het toch om een geheel nieuwe trend?

In de 'SD Times' (Software Development Times, www.sdtimes.com) probeerde Graham Berkeley, e-business product management director van Compuware, de kreet 'IDE' te vermijden. Het heeft naar zijn mening te zeer 'a nasty 3GL ring to it'. Columnist Rick van der Lans sprak daarentegen onlangs van de 'wederopstanding van case'. Er is echter naar zijn mening wel sprake van een nieuw type IDE dat wellicht ook het gebruik van een nieuwe naam rechtvaardigt. Hij ziet wel iets in de kreet 'IDEA', wat zou moeten staan voor 'integrated development environment and analysis'. Kenmerkend voor dit soort producten is het feit dat de afstand die traditioneel bestaat tussen enerzijds het specificeren van een applicatie en de feitelijke programmacode nu eindelijk kan worden overbrugd. Dat hierbij nadrukkelijk geleund wordt op Rational's en door de OMG (Object Modelling Group) tot standaard verheven UML (Unified Modelling Language) zal duidelijk zijn.

> "De twee duidelijke groeiers zijn C# en – maar dan nog veel meer – Java"

De voordelen van een dergelijke aanpak – waarbij specificatie en implementatie nadrukkelijk van elkaar gescheiden blijven – zijn erg groot, meent Edwin Schumacher, 'director product management' bij Compuware in Amsterdam. Hier is het development center gevestigd waar OptimalJ is ontwikkeld. Het nieuwe Compuware-product is onder leiding van Schumacher gebouwd door een inmiddels 65 man sterk ontwikkelteam. "Doordat veel van de complexiteit van objectgeoriënteerd werken en de Java 2 Enterprise Edition (J2EE) architectuur wordt afgeschermd, zijn veel grotere aantallen programmeurs dan voorheen in staat om bedrijfskritische Java-applicaties te schrijven."

"We zijn ruim anderhalf jaar geleden aan de ontwikkeling van OptimalJ begonnen", vertelt Schumacher. "Wanneer je goed naar de ontwikkelingen in de markt kijkt, zie je enkele zeer interessante ontwikkelingen ontstaan. We hebben onder andere in kaart gebracht aan de hand van welke ontwikkelmodellen nieuwe applicaties worden gebouwd. Kijk je naar de cijfers en de verwachtingen die hierover onder andere door Gartner zijn gepubliceerd, dan zie je een duidelijke deling in de markt ontstaan. Java en Microsoft's .Net zullen in pakweg 2005 de

twee toonaangevende modellen vormen. De rol van de proprietary aanbieders van 4GL-, AS/400-, RPG- of Cobol-tools zal beduidend afgenomen zijn."

Deze trend is weergegeven in figuur 1. Dat dit directe gevolgen heeft voor de populariteit van programmeertalen zal duidelijk zijn (figuur 2). "Cobol zien we vrij sterk teruglopen, terwijl Visual Basic en C++ eveneens afnemen, al gaat dat een stuik minder snel. Veel programmeurs die zich tot nu toe op C++ hebben gericht, zullen waarschijnlijk overstappen op C#, terwijl veel Visual Basic-ontwikkelaars vermoedelijk richting een .Net-variant zullen gaan. De twee duidelijke groeiers zijn C# en – maar dan nog veel meer – Java."

HANDJEVOL De Java-lijn in figuur 2 gaat vrij stijl omhoog en komt rond 2005 uit op 2,5 miljoen. Dat wil zeggen dat er in dat jaar een verwachte behoefte aan Java-programmeurs bestaat ter waarde van dat getal. Daar zit een stevig knelpunt. Marktanalisten gaan er van uit dat we wereldwijd op dit moment niet eens zo vreselijk ver van dit aantal af zitten. Wat echter wel een probleem is, is het feit dat veel van de huidige Java-programmeurs nauwelijks iets afweten van bijvoorbeeld Java 2 Enterprise Edition ofwel J2EE. Schumacher: "En dat is natuurlijk wel de omgeving waar we het hier over hebben. Naar schatting kan een aantal van 1,75 miljoen uit de huidige populatie aan Java-mensen beter worden getypeerd als hobbyïst, student of op zijn best als een programmeur die – naast een andere taal of omgeving – ook iets met Java doet. Daarnaast zien we een groep van laten we ze maar 'qualified professional developers' noemen. Dat zijn ontwikkelaars die een behoorlijke kennis van Java hebben, maar dan toch vooral gericht op Java 2 Standard Edition (J2SE). Het aantal specialisten dat goed uit de voeten kan met J2EE schatten wij wereldwijd op minder dan dertigduizend. Als we dat vertalen naar de Nederlandse situatie hebben we het over slechts een handjevol."

Aangezien analisten als Gartner er van uit gaan dat Java en .Net weliswaar pure concurrenten zullen zijn, zal er toch wel degelijk sprake zijn van verschillen. Niet alleen zullen veel grote organisatie beide modellen in huis halen, de groei van Java zou bovendien wel eens vooral kunnen gaan komen van toepassingen voor enterprise-applicaties met duidelijke eisen ten aanzien van met name de schaalbaarheid, vertelt Schumacher. Met andere woorden: er is sprake van zowel een bedreiging – een tekort aan Java/J2EE-ontwikelaars – als een kans. Namelijk een duidelijk groeiscenario voor zware Java-applicaties.

Compuware denkt net als softwarebedrijven als Embarcadero of Togethersoft de dreiging voor een belangrijk deel te kunnen neutraliseren door een ontwikkelomgeving aan te bieden die het modelleren van een applicatie en het feitelijke programmeren combineert binnen één omgeving, maar tegelijkertijd beide activiteiten nadrukkelijk gescheiden houdt. Dat een dergelijke aanpak interessante voordelen kan bieden, zal duidelijk zijn. Ontwikkelaars kunnen zich hierdoor concentreren op de applicatie zelf, zonder dat zij diepgaande kennis van een programmeeromgeving nodig hebben. Bovendien kan deze aanpak de weg vrij maken voor meerdere implementaties ofwel het gebruik van meerdere programmeertalen.

De vraag is nu natuurlijk wel: hoe is deze manier van ontwikkelen bij OptimalJ mogelijk gemaakt? Kenmerkend voor OptimalJ is een geheel modelmatige manier

Active synchronization

Om het onderhoud van J2EE-bedrijfsapplicaties te kunnen vergemakkelijken en versnellen, beschikt OptimalJ over een zogeheten 'active synchronization'. Deze faciliteit zorgt ervoor dat wijzigingen op model-niveau automatisch worden doorgevoerd tot in de code.

OptimalJ is gebaseerd op OMG's Model Driven Architecture (MDA) en onderscheidt derhalve drie modelniveaus: een Platform Independent Model (PIM), in OptimalJ heet deze het Domain Model, een Platform Specific Model (PSM), bij Compuware de application modellen voor Web, EJB en DBMS, en een Code Model, ofwel de uiteindelijke Java source code.

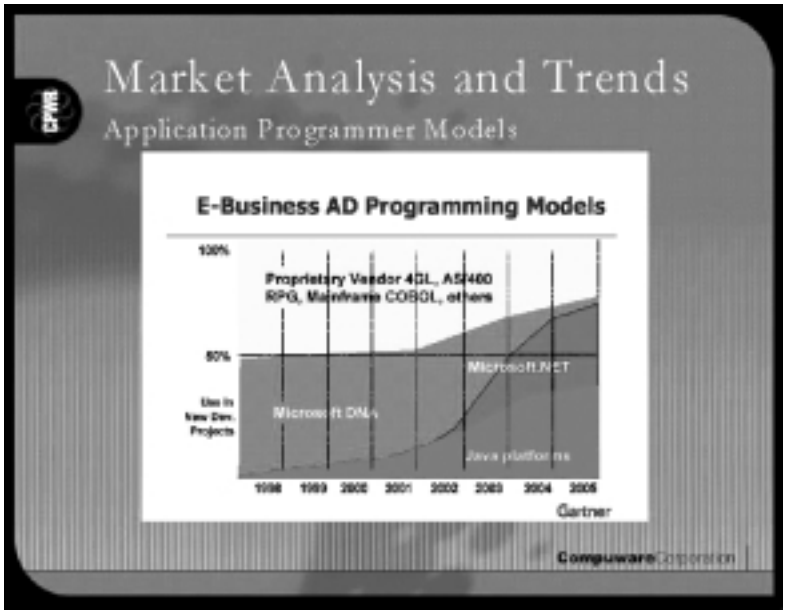
Wanneer bijvoorbeeld een wijziging wordt uitgevoerd in het Domain Model – denk aan het toevoegen van een klasse – zorgen de transformatie-patterns die verantwoordelijk zijn voor het vertalen van het Domain Model naar de drie submodellen (EJB, Web, DBMS) ervoor dat de juiste elementen worden toegevoegd aan deze modellen. In dit voorbeeld zal OptimalJ in het DBMS Model een nieuwe tabeldefinitie toevoegen. Het EJB Model wordt uitgebreid met een Entity Component-definitie en het Web Model wordt uitgebreid met een Web component-definitie.

Vervolgens zal een andere set van transformatie-patterns – namelijk die patterns verantwoordelijk zijn voor het vertalen van het PSM naar de code – ervoor zorgen dat de code wordt uitgebreid. In dit voorbeeld zullen JavaServer Pages en servlets worden gegenereerd voor de gebruikersinterface. Verder zal er een EntityBean worden gegenereerd met de nodige additionele klassen zoals een Helper klasse, een Primary Key klasse en dergelijke. Voor de nieuwe tabel zal een SQL-script worden gegenereerd dat een database administrator in staat stelt deze tabel in de database aan te maken.

Compuware noemt active synchonization een intelligent proces. Dat wil zeggen dat OptimalJ het verschil herkent tussen het genereren van nieuwe elementen, het wijzigen van bestaande elementen en het verwijderen van elementen. Hierdoor kan voorkomen worden dat de gehele applicatie simpelweg opnieuw gegenereerd wordt, maar alleen die delen die daadwerkelijk moeten worden aangepast.

De ontwikkelaar kan zelf bepalen op welk moment de synchronisatie plaatsvindt. Het kan automatisch gebeuren, waarbij iedere wijziging direct wordt doorgevoerd (real-time). Maar wanneer bijvoorbeeld veel wijzigingen ineens moeten worden uitgevoerd die ook nog eens aan elkaar gerelateerd zijn, kan de ontwikkelaar er ook voor kiezen om de synchronisatie later uit te voeren.

Tenslotte herkent active synchronization het verschil tussen gegenereerde code (guarded blocks) en handmatig ingebrachte code (free blocks). Tijdens het synchronisatieproces zal alleen gegenereerde code worden aangepast, zodat de handmatige code niet verloren gaat.



FIGUUR 1. De verwachtingen van adviesbureau Gartner ten aanzien van het toekomstig gebruik van de diverse applicatieontwikkelmodellen.

van werken. Van de eerste modelleringen tot de feitelijke ingebruikname van de applicatie gebeurt alles in principe 'model driven'. Daarbij wordt nadrukkelijk steun verleend aan de technieken en notaties van de Unified Modelling Language (UML).

ACTIVE SYNCHRONIZATION OptimalJ is gebaseerd op NetBeans en wordt daar als het ware over heen geïnstalleerd. Dat wil zeggen dat menu- en vensterstructuren gedeeld worden, terwijl de met OptimalJ gegenereerde code bovendien binnen NetBeans kan worden bewerkt. Overigens wordt ook Sun's implementatie van NetBeans – Forte – ondersteund. Aan ondersteuning van andere IDE's wordt gewerkt.

Een applicatie wordt OptimalJ opgezet door met de hulp van een grafische editor een Domain Model met de bekende business classes en dergelijke te creëren. Ook is het mogelijk bestaande modellen of modellen die met een separaat modelleringhulpmiddel zijn gecreëerd binnen de werkomgeving te halen. Dit gebeurt via een UML/XMI-importfaciliteit (XML Metadata Interface) waarbij het UML-model als een XMI-bestand wordt geïmporteerd om vervolgens als Domain Model binnen OptimalJ te worden gebruikt en eventueel verder bewerkt. Bestaande modellen kunnen bovendien via JDBC (Java Database Connectivity) uit een relationele database omgeving worden geïmporteerd. Het omzetten naar de objectgeoriënteerde omgeving gebeurt hierbij automatisch door gebruik te maken van mapping patterns.

> Kenmerkend voor OptimalJ is een geheel modelmatige manier van werken

Het is mogelijk te werken met dynamische business rules. Deze worden als vastgelegd in een 'runtime rules base' en worden op dynamische wijze geïnterpreteerd. Dat biedt als voordeel dat rules aangepast kunnen worden, zonder dat de code moet worden aangepast. OptimalJ beschikt hiertoe over een business rules editor, die overigens ook met statische rules uit de voeten kan.

Om van het Domain Model van de applicatie te kunnen komen tot de uiteindelijke code zijn drie submodellen nodig: voor de business logica (EJB Model), voor de communicatie met de database (DBMS Model) en voor de presentatie (Presentation of Web Model). Schumacher: "Ieder submodel bevat alle definities van de componenten die nodig zijn om de gewenste functionaliteit te kunnen implementeren. Vanuit deze submodellen kan de uiteindelijke implementatie worden gegenereerd: JavaServer pages, servlets, Enterprise JavaBeans en SQL-scripts."

De drie submodellen kunnen separaat maar desgewenst ook tegelijk worden gegenereerd. Bij wijzigingen zorgt een faciliteit die 'active synchronization' wordt genoemd er voor dat het Domain Model en de submodellen op elkaar afgestemd blijven. De synchronisatiefaciliteit maakt het mogelijk dat de submodellen automatisch alle relevante definities uit het Domain Model erven. Hierbij wordt het gebruik van 'guarded blocks' en 'free blocks' met Java-code ondersteund, zodat ook bij het opnieuw genereren van code eigen aanpassingen in de Java-code behouden blijven. De actieve synchronisatie biedt echter geen – wat wel genoemd wordt – 'round-trip engineering'. Het is dus niet mogelijk om vanuit bestaande code een model te genereren.

JAVACENTRAL Bij het genereren van de uiteindelijke code wordt bovendien gebruik gemaakt van 'patterns' zoals deze bijvoorbeeld door Sun zijn gepubliceerd. Vooralsnog is bij OptimalJ echter niet voorzien in een patterns editor, al kunnen patterns natuurlijk wel via business rules of via free blocks worden aangepast. Het achterwege laten van een dergelijke editor is een bewuste keuze voor de eerste versie van het product geweest. Het werken met free code blocks of het aanpassen van standaard patterns is in de visie van Schumacher namelijk alleen weggelegd voor het eerder genoemde handjevol echte J2EE-ontwikkelaars.

Dit is een aspect waar vanuit Compuware de nodige aandacht aan zal worden besteed. Het gaat namelijk om een vrij lastig punt waarbij wellicht zelfs van een cultuuromslag sprake moet zijn. Dat heeft met name te maken met de redelijk eigenwijze houding van veel erva-

ren J2EE-ontwikkelaars dat alleen de eigen code goed is. Sterker nog: vaak lijkt de houding te worden aangehangen dat alleen de eigen code goed genoeg is. Voor hen zou een patterns editor dus zeker gewenst zijn. Veel programmeurs die met OptimalJ aan de slag gaan, zullen echter gezien de positionering die Compuware heeft gekozen niet over de kennis en kunde beschikken om zelf beslissingen te nemen over de vraag welke Java-code nu precies gegenereerd wordt. Een patterns editor is dan ook niet direct noodzakelijk – of misschien zelfs wel niet gewenst – voor die programmeurs die nadrukkelijk gebruik willen maken van de scheiding tussen model en codegeneratie, maar maakt het product wel interessanter voor J2EE-ontwikkelaars die wél invloed op de gegenereerde code willen kunnen uitoefenen.

In dit spanningsveld is een belangrijke rol weggelegd voor JavaCentral (http://javacentral.compuware.com). Deze website is bedoeld als een community voor OptimalJ-ontwikkelaars. Schumacher: "De huidige patterns zijn over het algemeen gericht op het genereren van code. Ik kan me echter voorstellen dat er in een dergelijke community ook patterns ontstaan die meer business-georiënteerd zijn. Maar denk ook aan functiegerichte patterns, bijvoorbeeld bedoelt voor het regelen van autorisaties of het realiseren van een emailfaciliteit binnen een applicatie. Zo zijn er natuurlijk honderden patterns te bedenken. Die kun je zelf binnen je eigen organisatie ontwikkelen, maar wanneer dit gebeurt via een community waarin patterns beschikbaar worden gesteld en dus door andere hergebruikt kunnen worden is dat wellicht wel zo interessant."

Als geïntegreerde ontwikkelomgeving zijn vooralsnog alleen NetBeans en Forte beschikbaar. Op de standaard voorzieningen van NetBeans heeft Compuware echter wel uitbreiding gemaakt. Het gaat om een multidocument-interface, waardoor bijvoorbeeld een duidelijk onderscheid mogelijk is tussen guarded en free blocks. Schumacher:

Drie versies

OptimalJ is beschikbaar is drie versies: de gratis Community Edition, de Professional Edition met OptimalServer als optie en de Business Edition die tevens faciliteiten kent voor workflow (OptimalFlow) en message translation (OptimalBridge). Toekomstige uitbreidingen van OptimalJ die nu reeds door Compuware zijn aangekondigd, omvatten onder andere een webservices-implementatie, een pattern editor, bredere ondersteuning van IDE's en een nadrukkelijker integratie met de application lifecycle-producten van het softwarebedrijf. Denk aan de DevPartner Java Edition ontwikkeltools, de QACenter E-Business Edition en QACenter Performance Edition reeks van testhulpmiddelen en EcoSystems voor applicatiebeheer.

FIGUUR 2. De ontwikkeling van het marktaandeel van een aantal programmeertalen.

"Aangezien NetBeans een open source-project is, hebben we deze MDI-ontwikkeling op onze beurt weer aan de open source-wereld teruggegeven."

ERVARINGEN Hoewel OptimalJ al enige maanden beschikbaar is, zijn er nog weinig reacties van eerste gebruikers bekend. Schumacher noemt onder andere autofabrikant Ford als een partij die met het product experimenteert. Daarnaast zijn enkele ervaringen bekend uit Frankrijk. Hier heeft een regionaal ziekenhuis – CHR Lille – met OptimalJ gewerkt bij de herbouw van een deel van een legacy-applicatie. Voorheen gebeurde dit met een niet nader genoemde Java IDE, aldus project manager Yves Beauchamp. Men werkt nu met OptimalJ en verwacht een flinke tijdwinst te kunnen boeken.

Ook analisten zoals de Hurwitz Group zijn positief over OptimalJ, met name over het werken met patterns die gelden als 'best practices coding templates'. Een applicatie kan hierdoor sneller ontwikkeld worden. Ook standaarden als UML, MOF (Meta Object Facility) en dergelijke worden ondersteund.

Omdat vooralsnog maar heel weinig echte J2EE-applicaties in gebruik zijn genomen, zijn analisten het er over eens dat het voor de acceptatie van OptimalJ van groot belang is om met een overtuigende 'business case' te komen. Deze kan andere organisaties inzicht geven in de mogelijkheden en potentiële valkuilen van het gebruik van dit soort tools, maar vooral ook de redelijk ingrijpende overstap naar een model-based manier van ontwikkelen van Java-applicaties.

Robbert Hoeffnagel