Oracle en MDA

deel 2

In het eerste deel zijn we ingegaan op MDA en de rol van Oracle Designer hierin. In dit tweede deel plaatsen we wat kanttekeningen bij de succesfactoren uit het eerste deel, behandelen we de modelgedreven ontwikkelmogelijkheden in JDeveloper en Jheadstart, de rol van MDA daarin, de vraag in hoeverre JDeveloper/JHeadstart Oracle Designer als modelgedreven ontwikkeltool kan opvolgen én de vraag hoe bedrijven met veel Designer specialisten nu kunnen instappen in de Oracle Java ontwikkellijn.

Deel één van dit artikel is gepubliceerd in Optimize 6/2007 en ging in op de positie van de twee Oracle ontwikkellijnen, vervolgens is het MDA concept aan bod geweest, waarna de succesfactoren van het modelmatige ontwikkelen het artikel afsloot.

Kanttekeningen bij modelgedreven ontwikkelen

Als modelgedreven ontwikkelen alleen voordelen zou hebben zoals in het eerste deel van dit artikel is beschreven dan zou modelgedreven ontwikkelen veel populairder zijn.
Modelgedreven ontwikkelen heeft echter ook nadelen:

- Modelgedreven ontwikkelen kan niet zonder goede tooling. Goede tooling maakt flexibel ontwikkelen mogelijk. Gebleken is echter dat er een vrij lange leercurve nodig is, wil de engineer de tool goed onder de knie krijgen. Deze leercurve wordt als een nadeel ervaren in onze snel veranderende ICT wereld.
- Het werken met geautomatiseerde transformatieslagen heeft, hoe uitgebreid de besturing van deze transformatie ook is, altijd beperkingen. Deze beperkingen uiten zich bijvoorbeeld in de (on)mogelijkheden om de layout van een userinterface vorm te geven. Dit wordt door gebruikers wel eens als een beperkende factor gezien.
- Het modelleren van alle relevante specificaties in de grafische modellen blijkt niet mogelijk te zijn: in een ERD kunnen bijvoorbeeld lang niet alle business rules aangegeven worden. Ook de sturing in welke laag een rule geïmplementeerd dient te worden, is niet in een model aan te geven. UML biedt meer mogelijkheden en met versie 2.0 lijkt er alweer een stap extra gezet te zijn. Ook het toepassen van UML-profielen kan enig soelaas bieden als de UML-modellen zelf te kort schieten. Nog steeds kunnen echter niet alle specificaties in een model gezet worden en zijn aanvullende middelen noodzakelijk. Dit kan in de vorm van een repository die op verschillende manieren gerepresenteerd kan worden (property pallets, object navigators: in/uitklapbare directory-structuren, of op grafische wijze). Deze aanvullende middelen worden vrijwel altijd tool-specifiek geïmplementeerd en de uitwisselbaarheid boet dan duidelijk aan kracht in.
- Doordat er op eenvoudige wijze gebruik gemaakt kan worden van werkende voorbeelden als prototype, bestaat bij de gebruikers, maar ook bij het management de indruk dat het systeem al bijna af is. Prototypes zijn echter het product van een eerste generatieslag en vormen als het ware een globale voorstelling van hoe het systeem gaat werken. Dat juist in de afwerking de meeste tijd zit, is dan ook moeilijk uit te leggen.
- Een van de randvoorwaarden is de inrichting van versiebeheer. Versiebeheer op de modellen zelf en de transformatiegegevens blijkt in de praktijk een complexe activiteit te zijn: het te ontwikkelen systeem bestaat in elke fase uit vele verschillende componenten, die elk herkenbaar vastgelegd dienen te worden. Initieel zal er één versie van elke component bestaan. Na een aantal tranformatieslagen zullen er echter al snel meerdere versies van de componenten bestaan. Daarnaast zal een set van componenten van een bepaalde versie bijvoorbeeld voor testdoeleinden opgeleverd gaan worden. Al snel zullen door herstelacties, aanvullende / vervangende sets opgeleverd worden en dan betreft het niet alleen de gegenereerde code.

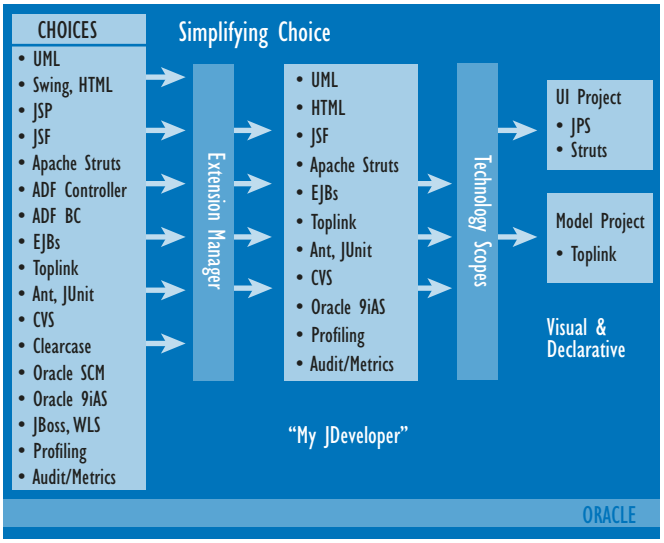
JDeveloper/ADF /JHeadstart

Als Oracle JDeveloper positioneert als de opvolger van Designer, hoe zit het dan met modelgedreven ontwikkelmogelijkheden? Dit hoofdstuk tracht hier meer duidelijkheid over te verschaffen.

JDeveloper/ADF

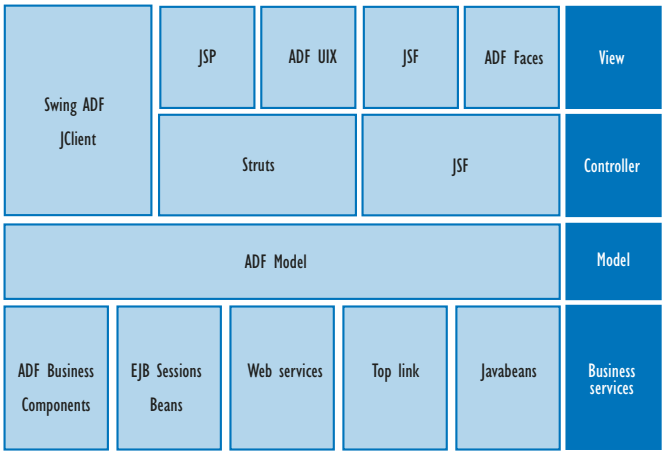
In eerste instantie was JDeveloper de IDE voor Java-programmeren. De kenner zal kunnen constateren dat JDeveloper inmiddels veel meer is dan dat. Oracle presenteert JDeveloper als een tool met keuze: ontwikkelaanpak, deploymentplatform en technologiekeuzes/frameworks zijn door middel van een 'extension manager' te kiezen. Zie figuur 1.

Door middel van het ADF (Application Development Framework) maakt Oracle het mogelijk gericht J2EE-conforme systemen te ontwikkelen, waarbij MVC (Model, View, Controller) als standaard ontwikkelpatroon gekozen is. Vanuit de modelgedreven ontwikkelaanpak zijn uiteraard de UML modellers interessant om te kiezen als ontwikkelhulpmiddel.



Figuur 1: JDeveloper keuzemenu (bron Oracle)

Met ADF ligt in JDeveloper de focus op J2EE en het MVC. Oracle heeft voor deze architectuur de volgende invulling opgezet:



Figuur 2: technology stack ADF

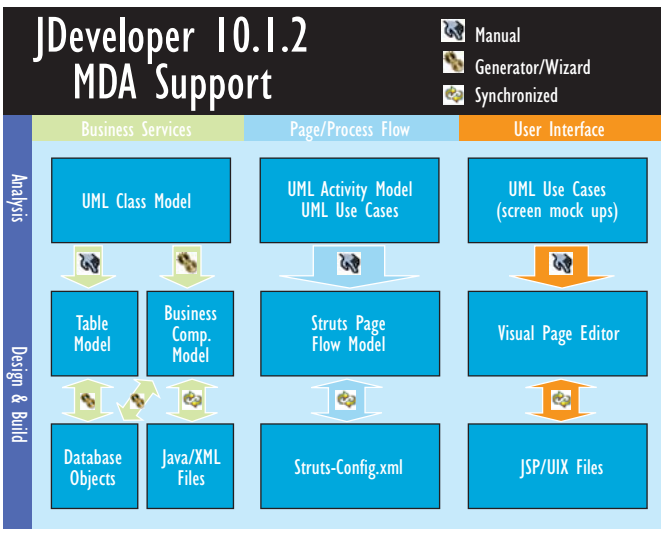
Nu Java in de internationale ICT-wereld steeds meer aan populariteit wint, zijn de activiteiten in deze technologierichting ook zeer sterk toegenomen. De technologie is nog sterk in beweging, wanneer het ene framework net gelanceerd is als 'nieuwe' standaard, is de volgende alweer bedacht. Verder valt op dat de complexiteit van de frameworks op zichzelf, laat staan in combinatie met elkaar, erg groot is.

In JDeveloper is het al geruime tijd mogelijk enkele 'low level' UML modellen op te nemen (activity en class diagram). Vanuit deze modellen kan code gegenereerd worden en vice versa. Inmiddels zijn er nog een aantal nieuwe UML modellers toegevoegd (sequence en use case).
JDeveloper ondersteunt nu de volgende modellen in de volgende MDA-stappen (ontwikkelfasen):

PIM (Uppercase, Analysis):
- UML Class Model
- UML Use Case Diagram
- UML Activity Diagram
- UML Sequence Diagram (vanaf versie10.1.3)

PSM (Lowercase, Design):
- UML Database Diagram
- UML Business Components Diagram
- UML Java Class Diagram
- UML EJB Diagram
- UML Web Services Diagram
- UML Struts Page Flow Diagram
- Visual Page Editor

In hoeverre JDeveloper nu de transformatieslagen ondersteunt, wordt in onderstaand schema duidelijk gemaakt.

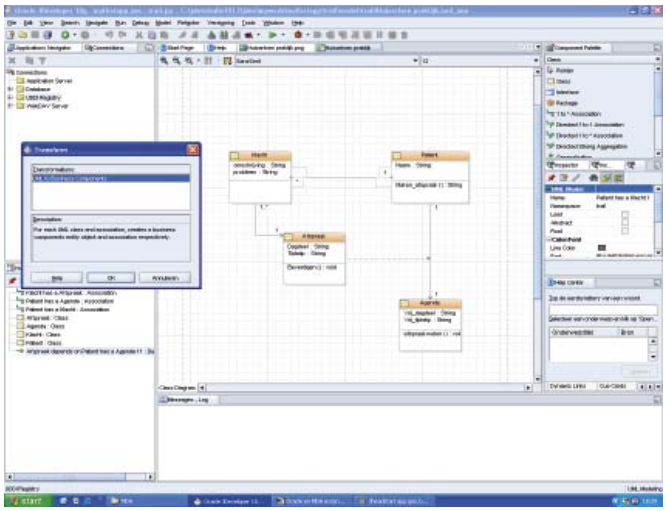


Figuur 3: JDeveloper MDA support (bron Oracle)

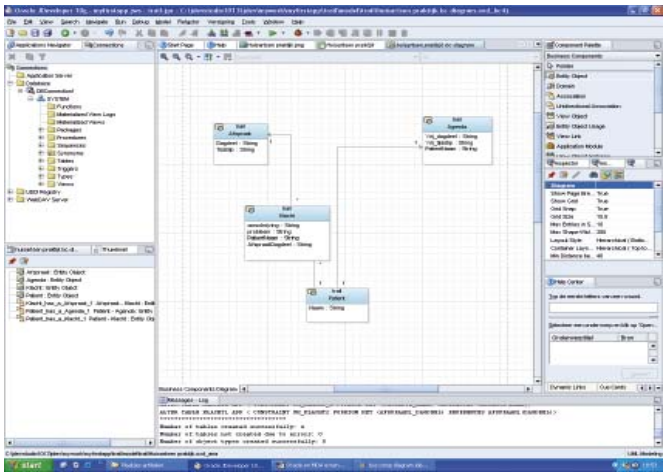
Uit bovenstaand schema is op te maken dat de transformaties vanuit analyse-niveau (Platform Independent Model: PIM) nog maar gedeeltelijk ondersteund worden door wizzards.
Op dit moment geeft Oracle aan dat ze in de volgende release de huidige wizzard verbeteren en er nog een tweetal toevoegen: vanuit het class diagram naar het databasediagram (table model) en
vanuit het activity model naar het Page flow model.

Hoe deze modellen onderling op hetzelfde niveau consistent gehouden worden, is niet duidelijk. Overigens wordt bij de algemeen geldende MDA-gedachte (OMG) daar ook weinig aandacht aan geschonken.

Hieronder een voorbeeld hoe er vanuit een class diagram ADF Business Components gegenereerd worden:



Voorbeeld 1: Het class diagram met de tranformer popup voor de ADF Business Components

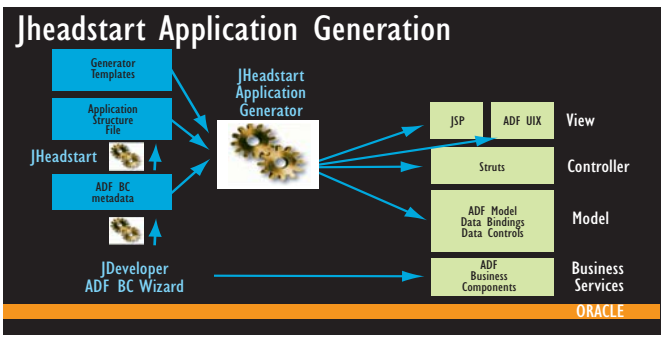


Voorbeeld 2: de getransformeerde classes naar Business Components in het Business Components Diagram

JHeadstart

Zonder JHeadstart te gebruiken, is men in staat volgens MDA (of liever gezegd modelgedreven) te ontwikkelen. Momenteel draait JDeveloper nog naast Designer/Forms als ontwikkelplatform voor de Java-technologie. Het mag duidelijk zijn de meeste aandacht van Oracle naar JDeveloper gaat. Met Designer/Forms heeft Oracle een flink marktaandeel in de administratief ondersteunende systemen. JDeveloper als Java EDI, maar zeker als modelgedreven ontwikkeltool, biedt niet genoeg tegenwicht om met eenzelfde productiviteit als Designer/Headstart op hetzelfde marktsegment succesvol te zijn. Het antwoord van Oracle consulting Nederland is dan ook het product JHeadstart, dat de productiviteit voor het ontwikkelen van administratief ondersteunende systemen aanzienlijk vergroot en tevens een migratiebrug slaat tussen Designer en JDeveloper. De met wizzards in JDeveloper ontworpen user interfaces, bieden nog niet alle mogelijkheden, die een de schermen inForms te bieden hebben (bijvoorbeeld List of values, multi row insert, quick search en verschillende parent-child lay-out stijlen). Dergelijke 'features' dient men er handmatig bij te coderen en dit gaat dan ten koste van de productiviteit. Zoals ook bij Oracle Designer productiviteit een belangrijke rol speelde, zagen de makers van JHeadstart ook wel mogelijkheden om de productiviteit van JDeveloper/ADF te verbeteren. JHeadstart biedt een verdere ondersteuning in de declaratieve ontwikkelwijze, maakt prototyping mogelijk en is volledig geïntegreerd in de JDeveloper tool.

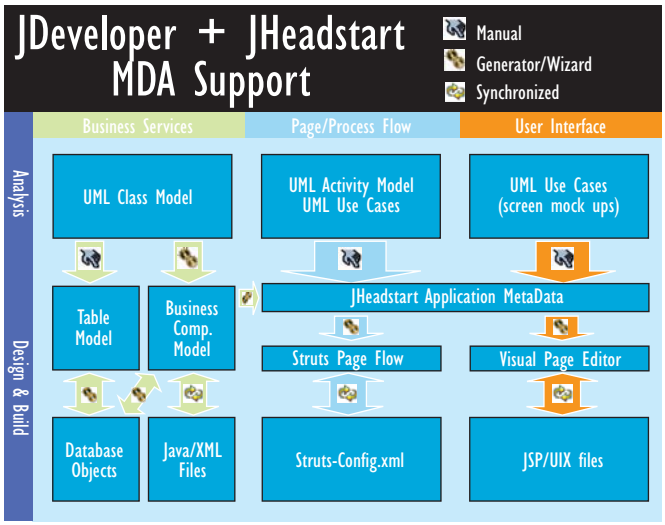
In onderstaand schema zijn de extra generatiemogelijkheden van JHeadstart aangegeven:



Figuur 4: JAG: JHeadstart Application Generation (bron Oracle)

De JAG genereert geen Java-code, maar herbruikbare componenten op design-niveau. De JAG wordt aangestuurd vanuit:
- de op XML gebaseerde 'Application Structure File'
- de ADF Business Components op design-niveau, in de vorm van 'properties'

Het schema dat aangaf in hoeverre JDeveloper MDA ondersteunt, kan in combinatie met JHeadstart als volgt verder uitgebreid worden:



Figuur 5: JDeveloper/JHeadstart MDA support (bron Oracle)

Wederom is de transformatie vanuit PIM naar PSM gedeeltelijk ondersteund. Dit lijkt ook wel logisch, daar alle energie die in JHeadstart gestoken is, uitsluitend op het PSM en implementatievlak gericht is. Nu lijkt het dat JHeadstart het statement dat al eerder in dit artikel gemaakt is betreffende de complexheid van het ontwikkelen van J2EE-systemen, alleen maar meer waar te maken. De schijn bedriegt echter: alle extra generatieslagen nemen veel handmatig werk uit handen. Impliciet maken de generatieslagen een aantal keuzes, die anders door de engineer gemaakt moeten worden. Voor al die keuzes is veel kennis van J2EE nodig. Overigens is Oracle wel van plan JHeadstart uit te breiden met een 'customizable generator'. Hierbij leveren zij dan ook een XSLT stylesheet die als template gebruikt wordt om de generator van defaultwaarden te voorzien.

Conclusie

JDeveloper met ADF kan als een implementatie van MDA gezien worden. Hiervoor zijn een aantal 'meetpunten' en invullingen in JDeveloper op een rij gezet:

- De verschillende modellen zoals die in MDA voor dienen te komen (PIM, PSM, Implementatie code) zijn duidelijk te onderscheiden en UML wordt reeds met vier verschillende grafische modelleermogelijkheden ondersteund. De transformatieslagen zijn nog niet allemaal geautomatiseerd.
- Binnen de Java-technologie biedt JDeveloper/ADF diverse keuzes van frameworks en de daarbij horende PSM. Een engineer kan gebruik maken van algemeen toepasbare componenten die gebaseerd zijn op open standaarden. Herbruikbaarheid en uitwisselbaarheid van componenten is hiermee gewaarborgd. Ook kan het PIM gebruikt worden voor een PSM in een andere technologie, dit zal dan echter niet in JDeveloper/ADF zijn.

Indien men de invulling van de concepten die achter MDA zitten bekijkt, met name het aspect of een tool modelgedreven ontwikkelen mogelijk maakt, dan kan dit positief beantwoord worden. Hierbij is wel een kanttekening te maken. Met louter alleen JDeveloper/ADF zijn niet alle transformaties geautomatiseerd. Diverse componenten dienen nog met de hand gemaakt te worden. Hierbij biedt JDeveloper wel een declaratieve ontwikkelwijze, welke de productiviteit toch verhoogt.
Met JHeadstart als add-on komt het modelgedreven ontwikkelen pas echt tot zijn recht. JHeadstart biedt echter alleen meerwaarde als een systeem een relatief groot aantal user interfaces bevat welke sterk gericht zijn op gegevensverwerking en manipulatie (bijvoorbeeld bij administratief ondersteunende systemen). De mogelijkheden van het verrijken van de userinterface met behulp van JHeadstart is dan ook sterk op gegevensverwerking gericht.

Is JDeveloper/ADF, eventueel aangevuld met Jheadstart, nu een MDA-conform ontwikkeltool? Het antwoord hierop is 'nee'. Het is mooi als je een tool vanaf 'scratch' bouwt, om dan MDA als leidraad te gebruiken. Echter JDeveloper is een tool die al reeds jaren bestaat en daarmee ook een eigen historie heeft. Als Oracle hun eigen lijn in de ontwikkeling van de tool blijft volgen, waardoor niet alle MDA-concepten ingevuld zijn, is daar ook wat voor te zeggen. Zie verder ook 'de positie van MDA': het belangrijkste is dat een ontwikkeltool in de praktijk goed werkbaar is en daar lijkt Oracle al goed mee op weg te zijn.

Met betrekking tot de opgedane ervaring in modelgedreven ontwikkelen kan gezegd worden, dat deze ook gelden voor de ADF/JHeadstart combinatie: de productiviteit zal door de generatoren zeker omhoog gaan. Ook bij het ontwikkelen met ADF/

> ***Alle extra generatieslagen nemen veel handmatig werk uit handen***

JHeadstart zullen er goede roadmaps, standaards en richtlijnen nodig zijn. Het gevaar van een verkeerde of minder efficiënte weg inslaan is hier wellicht nog groter dan bij het ontwikkelen met behulp van Designer. Met de Designe/Headstart ervaring in hun achterzak, waarbij CDM (Custom Development Method)

uitgegroeid is tot de defacto standaard voor deze ontwikkelomgeving, is Oracle Nederland nu ook bezig een versie voor de JDeveloper/ADF/JHeadstart op te zetten: JCDM/OUM (Java Custom Development Method, die waarschijnlijk Oracle Unified Method gaat heten). Verder zorgen generatoren ervoor dat software snel voorhanden is, wat prototyping met werkende voorbeelden mogelijk maakt.

De genoemde kanttekeningen die al bij 'modelgedreven ontwikkelen' al vermeld zijn, blijven ook bij JDeveloper/ADF/ JHeadstart gelden. JDeveloper beschikt niet over een repository. In hoeverre dit nadelig is en welke alternatieven Oracle voor JDeveloper heeft ingezet is niet onderzocht.

Wat zit er in voor Designer ontwikkelaars?

Of een organisatie die veel energie gestoken heeft in Designer/ Headstart ontwikkellijn, nu over moet stappen naar de (Oracle) Java-lijn, is een vraag die in een breder perspectief bekeken dient te worden. Hierbij dienen de volgende zaken onderzocht te worden:

- Levensduurverwachting van bestaande systemen
- Strategische keuzes voor leveranciers en ontwikkelplatformen
- Op stapel staande nieuwe ontwikkelingen m.b.t. te ontwikkelen systemen, architecturen (bijvoorbeeld SOA)
- Kosten investering in nieuwe ontwikkelplatformen
- Omscholingstrajecten voor medewerkers
- Huidige kennis van de medewerkers

In de context van dit artikel is de overgang voor de medewerkers wel een nader te beschouwen aspect: hoe kunnen Designer/ Forms specialisten omgeschoold worden en wat betekent dit voor de kennis van een dergelijke specialist, moet hij/zij nu weer van onderaf aan beginnen? Hierover kan het volgde gezegd worden:

- Ontwikkelaars met kennis en ervaring in modelgedreven ontwikkelen hebben een flinke voorsprong: het concept zit al tussen de oren. Zoals al eerder in dit artikel is geconstateerd, vergt modelgedreven werken een lange leercurve. Voor Designer-specialisten is de overstap naar JDeveloper op dit punt een stuk gemakkelijker en zij zullen in de combinatie van JDeveloper/ADF/JHeadstart ook veel herkennen uit hun Designer/Headstart-achtergrond.
- Met de toevoeging van JHeadstart biedt JDeveloper de mogelijkheid om systemen te ontwikkelen zonder echt veel kennis van Java te hebben. Vooral door de technologie-wijziging (OO, UML, Java, J2EE etc) is de stap vanuit een ERD /SQL /relationeel gerelateerde omgeving groot. Zoveel mogelijk code genereren en meer aandacht voor de functionele eisen van een systeem maakt deze stap minder groot.
- De functierollen die in een op Designer gebaseerde ontwikkelomgeving bestaan, spelen ook mee in de overgang. Zo worden er de volgende functierollen onderkend in de Designer-omgeving:
- (Informatie)analist: is toolonafhankelijk maar gebruikt Designer als modelleertool
- Ontwerper met het ontwerp in Designer als primair aandachtsgebied, generatie van design (PSM naar code) secondair aandachtsgebied
- software-engineer, Generatie van Design (PSM naar code) als primair aandachtsgebied, ontwerp (uppercaselaag) als secondair aandachtsgebied.

Bovenstaande rollen zijn ook in een JDeveloper/ADF/ Jheadstart-omgeving te onderkennen. Een software engineer heeft verreweg het meeste baat bij een Designer-achtergrond: hij/zij kan wat meer leunen op de generatieprocessen en zich de nieuwe Java technologie langzamerhand eigen maken. Wanneer nieuwe projecten zich aandienen waar de te gebruiken toolstack de combinatie JDeveloper/ADF is, eventueel aangevuld met JHeadstart, kan voor een volgende samenstelling gekozen worden:

> ***Ontwikkelen met een modelgedreven ontwikkeltool vereist echter wel specialisatie***

- (Informatie)analist: is toolonafhankelijk maar gebruikt UML als modelleertaal (in JDeveloper)
- Ontwerper met ontwerp in JDeveloper (UML) als primair aandachtsgebied, generatie van design (PSM naar code) secondair aandachtsgebied
- Software engineer met JDeveloper /ADF/(eventueel JHeadstart) als specialiteit (Designer/Headstart achtergrond), voor het neerzetten van 'first cut' functionaliteit
- Software engineer met Java als specialiteit: voor het kiezen van Java / J2EE georiënteerde frameworks en de invullingen daarvan.

De beide software-engineers kunnen van elkaars kwaliteiten leren en kennis opdoen, zodat er naar verloop van tijd een betere mix van deskundigheid bij de verschillende rollen ontstaat.

Ontwikkeling van Java-systemen is niet gebonden aan een bepaald tool. Ontwikkelen met een modelgedreven ontwikkeltool vereist echter wel specialisatie. Designer richt zich op informatiegedreven systemen (administratief ondersteunende systemen). JDeveloper/ADF/JHeadstart als MDA tool lijkt hier-

Referenties

- The MDA guide version 1.01, OMG 2003
- MOF core specifications version 2.0, OMG, 2006
- Transforming software development: An MDA road map, Thomas Meservy /Kurt D. Fernstermacher, 2005
- Softwarekwaliteit van MDA tools E. De Groot, afstudeeropdracht, Universiteit Amsterdam, centrum voor wiskunde en informatica, 2005
- Risicobeheersing bij toepassing van MDA, Maarten de Vos, afstudeeropdracht, Open universiteit, Business processes and ICT, 2005
- UML modeling and MDA in Oracle JDeveloper 10G, statement of direction, Oracle 2004
- Oracle presentatie over J2EE/JDeveloper/ADF en JHeadstart, door Steven Davelaar, Technisch manager Oracle Nederland, 2005
- Oracle JDeveloper 10G (10.1.3) Overview, Oracle white paper 2005.
- Atos Origin presentatie: 'Boosting (Java) development productivity', door Kees Kranenburg 2004

voor een goede opvolger te zijn waarbij productiviteit ook weer een belangrijk verkoopargument is. Specialisatie in deze tooling is dan ook vanuit de Oracle-ontwikkelgemeenschap een logische keuze.

Geert Franssen is werkzaam bij Atos Origin als Technical Oracle Consultant, waar hij jarenlange ervaring heeft opgedaan in systeemontwikkeling met Oracle tools in verschillende grote projecten.

Noten

1 De genoemde rollen zijn niet uitputtend, zo zal er bijvoorbeeld ook een DBA, systeembeheerder en een configuratiemanager te onderkennen zijn. Echter de rollen informatieanalist, ontwerper en software-engineer zijn rechtstreeks werkzaam in het modelgedreven ontwikkelproces.