# Traditioneel of J2EE

## *De keuze in 2003*

***Veel bedrijven worstelen nog altijd met de beleidskeuze te (blijven) investeren in Oracle's traditionele producten voor maatwerk systeemontwikkeling, of inmiddels te kiezen voor het Java 2 Enterprise Edition (J2EE) platform. Opvallend hierbij is dat de argumenten die in de praktijk voor een bepaalde keuze worden gehanteerd vaak gebaseerd zijn op achterhaalde informatie. In dit artikel licht Harold Gerritsen de inhoud toe van Oracle's huidige marketing boodschappen, geeft hij informatie voor een zuivere afweging tussen ontwikkeling met de traditionele (Designer/Developer) productlijn en J2EE, en geeft hij een aantal aandachtspunten voor bedrijven die op de traditionele productlijn willen blijven.***

Vroeger, zo'n tien, twaalf jaar geleden, was het leven simpel voor ICT-managers die strategisch voor Oracle hadden gekozen. Met die keuze zat je immers safe. Goed, Oracle is nooit goedkoop geweest. Maar je wist wel dat er een dusdanig grote organisatie achter zat, dat je je nooit zorgen hoefde te maken over de continuïteit van de leverancier. En het productportfolio van Oracle was ook uitermate overzichtelijk. Een handvol producten (CASE*Designer, CASE*Dictionary, SQL*Forms, SQL*Menu, SQL*ReportWriter, SQL*Net en uiteraard de Oracle database, kennen we ze nog?), elk met een specifieke functionaliteit. De producten waren zelfs zo 'zelfstandig' dat vergelijkbare functies in verschillende producten er anders uitzagen, onder andere functietoetsen terug te vinden waren, en zelfs anders heetten. Dit was niet zo moeilijk voor te stellen overigens, omdat deze producten door verschillende development-afdelingen van Oracle werden ontwikkeld. Kennelijk had het destijds bij Oracle geen prioriteit om deze producten meer op elkaar af te stemmen.

Echter, ondanks het feit dat deze producten zo 'langs elkaar heen' werden ontwikkeld, kon je wel stellen dat er weinig overlap was in functionaliteit van de producten als geheel. Als je een specifiek automatiseringsvraagstuk had op te lossen met ontwikkeltools, was er voor elk type vraagstuk maar één ontwikkeltool bij uitstek geschikt. En eigenlijk kwam in die situatie geen verandering tot vijf jaar geleden. Inmiddels waren CASE*Designer en CASE*Dictionary geïntegreerd tot Designer/2000, SQL*Forms en SQL*Menu waren geïntegreerd tot Oracle Forms en vormde met onder andere Oracle Reports tezamen Developer/2000. Maar de functionaliteit van elk van de producten was tot die tijd eigenlijk vrij specifiek gebleven.

### De buitenwereld

Terwijl Oracle voortdurend bezig was zijn producten te integreren en innoveren, liep een nieuw fenomeen zich warm langs de zijlijn. Het begrip Component Based Development deed z'n intrede en, niet in de laatste plaats door Oracle's marketing zelf, het gebruik van het Internet en intra- en extranetten als platform voor applicatie-deployment nam een enorme vlucht. In het licht van deze ontwikkelingen heeft Oracle enkele slimme zetten gedaan. Om zijn relatieve achterstand op het gebied van CBD in te halen, heeft het eenvoudigweg producten ingekocht van andere leveranciers om deze vervolgens door te leveren onder 'eigen label'. Een voorbeeld hiervan is de Oracle Application Server. Oracle's eerste versie hiervan was in feite een label voor een aantal producten, zoals de Object Request Broker (ORB) van Visigenic voor de CORBA functionaliteit en de HTTP-listener van Spyglass voor de Webserver functionaliteit. Ook voor een tool om in Java te ontwikkelen heeft Oracle de moed (of is het wijsheid?) gehad in een ander softwarebedrijf zijn meerdere te erkennen. Softwareleverancier Borland had grote faam opgebouwd als producent van ontwikkeltools voor 3GL (startend met TurboPascal zo'n vijftien jaar geleden) en OO talen (onder andere

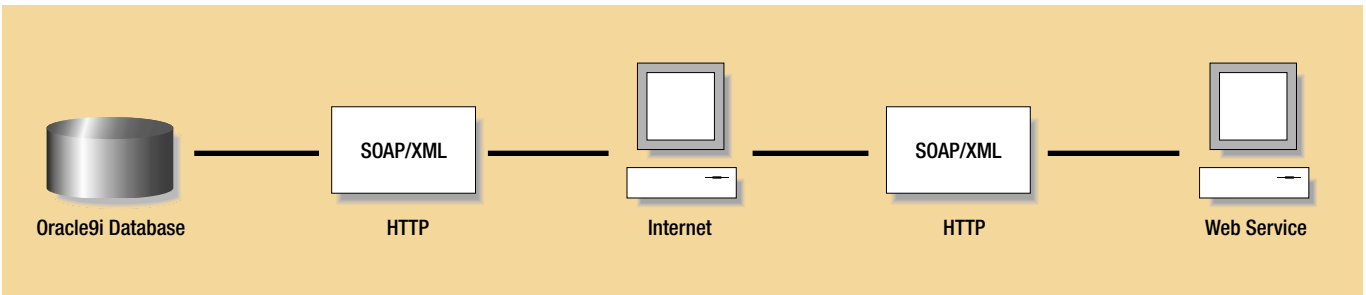
> ***Het leek er een beetje op dat Oracle zekerheid zocht door op twee paarden te wedden***

TurboC++, Delphi en JBuilder). Oracle heeft van Borland de Java ontwikkelomgeving JBuilder in licentie genomen, met recht op aanpassing en wederverkoop. Inmiddels is duidelijk geworden dat het Oracle hier uitsluitend ging om een goed 'startpunt' in te kopen. De actuele versies van JDeveloper zijn volledig opnieuw geschreven (JBuilder was oorspronkelijk in C geschreven, JDeveloper is gecodeerd in Java) en vriend en vijand zijn het erover eens dat de kwaliteit en functionaliteit van JBuilder niet in de schaduw kan staan van die van JDeveloper. Dit komt met name door het Business Components for Java (BC4J) framework dat automatisch met JDeveloper gegenereerd kan worden. Met dit framework kan de mapping van objecten uit de Java wereld op relationele tabellen in de Oracle database worden uitgevoerd en wordt een veelvoud aan design-patterns (zeg maar: standaard architectuur stijlen) geïmplementeerd.

## Twee paarden

Door al deze ontwikkelingen begon zo'n vier jaar geleden de situatie te ontstaan dat Oracle twee vergelijkbare productlijnen voerde. Het leek er een beetje op dat Oracle zekerheid zocht door op twee paarden te wedden. De situatie werd er voor de klanten van Oracle echter niet duidelijker op. Sterker nog: door de snelheid en eenzijdige marketingboodschappen van Oracle leek het er op dat er nog maar één productlijn bestond, namelijk de Java-lijn. Oracle ging meer en meer communiceren dat de uiteindelijke winnaar de Java-lijn zou zijn, en dat de traditionele productlijn afgebouwd zou gaan worden. Het hoogtepunt van deze richting was ruim twee jaar geleden, toen er op het gebied van de formele marketingstatements niets, maar dan ook helemaal niets meer viel te vernemen over de toekomst van Designer/Developer, terwijl er steeds meer geruchten gingen op seminars dat de traditionele lijn zelfs volledig gestopt zou worden. Daarbij kwam nog dat de productmanager van de Oracle Developer Tools in het Oracle Magazine schreef dat de projectnaam ('Project Cherokee') die werd gebruikt voor de ontwikkeling van de nieuwe versie van Oracle Developer, zou worden gereserveerd voor een ander project, namelijk voor het maken van een nieuwe Integrated Development Environment (IDE) gebaseerd op JDeveloper. Dit bericht, dat op een onopvallende plaats in een technisch artikel terug te vinden was, werd door vrijwel elke ontwikkelaar opgevat als de doodsteek voor Designer/Developer. Toen dit bericht doorsijpelde naar hogere echelons, trokken veel ICT-managers hun conclusies: stoppen met de traditionele productlijn en kijken naar alternatieven. Opvallend was hierbij dat klanten voor het eerst in jaren 'recalcitrant' gedrag gingen vertonen. De loyaliteit richting Oracle maakte plaats voor verontwaardiging over het feit dat men voor een voldongen feit werd geplaatst. Velen hadden het gevoel dat het consequente trouw zijn aan één leverancier door Oracle werd misbruikt om haar klanten een compleet andere productlijn op te dringen, met als gevolg verlies van gedane investeringen. Dat ging menig klant te ver. Hierdoor kozen steeds meer klanten de opstelling: 'Als ik dan toch voor Java moet kiezen, dan ga ik ook uitgebreid shoppen. En Oracle moet wel veel beter zijn dan andere alternatieven, wil ik de Oracle Java lijn gaan inzetten.' Toen deze teneur duidelijk werd, duurde het niet lang voor Oracle zich opnieuw op haar strategie ging beraden. Al gauw werd via het informele circuit, door losse uitlatingen op seminars, haar harde opstelling gerelativeerd. De eerste signalen van de kentering waren hoorbaar op de Oracle Developer Tools User Group conferentie in 2001. Oracle sprak zich hierbij uit over een dual-language strategie waarin zowel PL/SQL als Java zou blijven worden ondersteund, en zouden eveneens de repository en de developertools blijven worden ondersteund. Zoals de spreker, William Dwight, Vice President van Oracle Development Tools, zei: 'for as long as I live'. Veel klanten classificeerden deze uitspraken als 'zo zacht als boter'. Het kwam steeds vaker voor dat klanten eisten om een schriftelijke verklaring van de lengte van de supporttermijn van specifieke versies van de traditionele Oracle producten.

***Steeds meer klanten kozen de opstelling: 'Als ik dan toch voor Java moet kiezen, dan ga ik ook uitgebreid shoppen'***




*Afbeelding 1. Consumeren van een Web Service vanuit de database*

Daarom publiceerde Oracle in juli 2001 haar eerste 'Statement of Direction Oracle Forms'.

## The current Future Direction

Met de publicatie van de Statements Of Direction (SOD) begon een traditie die "Oracle's boterzachte uitspraken", zoals sommige klanten het verwoorden, wat harder maken. Het blijven niettemin statements met een bepaalde vrijheidsgraad. Ter illustratie: in een seminar over Oracle Designer 6i in het najaar van 2000 vroeg ik aan Ian Fisher van Oracle in welke richting het product zich in de toekomst zou gaan ontwikkelen. Zijn antwoord luidde: 'Our future direction….no, excuse me…. let me be more precise…..our current future direction (..)'. Hoe dan ook: door het bestuderen van de actuele SOD's krijgen we een goed beeld van de actuele marketingboodschap van Oracle. En, tot verrassing van vele klanten die ik op de inhoud van deze documenten wijs, is de geschiedenis waarin de traditionele productlijn ten dode was opgeschreven, volledig achterhaald. Voor een goed beeld van de huidige situatie zijn de volgende SOD's van belang: Oracle Forms[1], Migrating Forms Applications to J2EE[2] en Oracle Modeling Tools[3]. Enkele quotes uit deze SOD's zijn:

- "The Designer product is here to stay. Oracle is continuing to build on the Oracle9i stabilization release with enhancements to the Web PL/SQL generator, a new browser-based UI, and of course the continued emphasis on quality – all of these in successive upcoming maintenance releases."
- "Oracle is committed to continue developing, improving and supporting Oracle Forms."
- "Since the beginning, Oracle Forms has been a very important part of the technology stack of Oracle's own application suite – Oracle E-business Suite – which proves the level of productivity and scalability delivered by the product."
- "The strategy for upcoming Oracle Forms releases is to keep focusing on features that allow integration and interoperability between Oracle Forms and other systems or technologies."
- "Do I have to move to Java because Forms is de-supported? No. There may be many reasons to move to a Java platform but this should not be one of them. Oracle 9i Forms has over 20 new features and Oracle will continue to enhance Forms and continue to support its large install base."

*[1] Oracle Forms Statement of Direction – updated March 2003: http://otn.oracle.com/products/forms/htdocs/FormsSOD.html*

*[2] Migrating Forms Applications to J2EE: Statement of Direction – updated March 2003: http://otn.oracle.com/products/forms/htdocs/FormsJavaSOD.html*

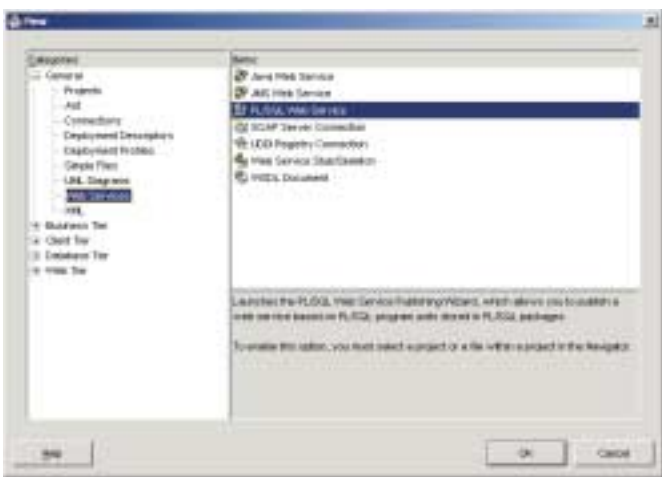
*[3] Oracle Modeling Tools: Statement of Direction – August 2002: http://otn.oracle.com/products/designer/htdocs/ModelingStrategy.htm*

In feite wordt in de SOD's de uitspraak gedaan dat Oracle zowel de traditionele productlijn als de J2EE productlijn in volle omvang zal blijven ondersteunen en doorontwikkelen. Er is dus wel degelijk een zuivere keuze mogelijk tussen de twee productlijnen en die is vrij van enige beïnvloeding door Oracle.

## De keuze

In het SOD document 'Migrating Forms Applications to J2EE' wordt het best beschreven welke factoren allemaal een rol spelen bij de keuze voor een van de twee productlijnen. De argumenten die worden genoemd om te kiezen voor Oracle Forms zijn:

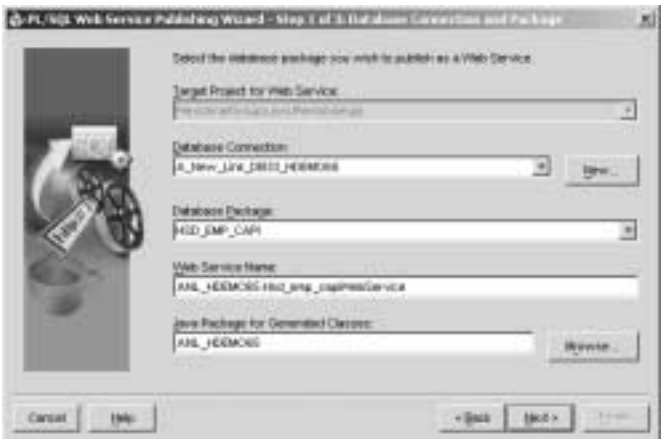
- een sterk productieve RAD aanpak gebaseerd op PL/SQL als scripting taal;
- een database-intensieve applicatie met data-entry en type-ahead;
- een applicatie met een rijke UI met ondersteuning van type-ahead, onmiddellijke validatie, auto-completion, en een rijke en productieve set van UI controls zoals tree controls, tabbed pages, LOV's etc.;
- deployment over het Internet, extranet of WAN met een redelijk voorspelbaar aantal eindgebruikers;
- sterke PL/SQL ontwikkel vaardigheden.



*Afbeelding 2a. Eerste stap, het starten van de Web Service publicatiewizard voor PL/SQL*

Argumenten die meer pleiten voor gebruik van de J2EE productlijn zijn:

- strategische wens vanuit de business om Java te gebruiken, met inherente voordelen zoals portability, flexibiliteit en het voldoen aan standaarden;
- een Self-Service of Business to Consumer applicatie;
- een applicatie gebaseerd op een ultra dunne HTML gebaseerd UI, waarschijnlijk gebaseerd op servlets of JSP's;



*Afbeelding 2b. Stap 2, het specificeren van bron en doel objecten*

- volledige flexibiliteit in deployment en mogelijkheden om elk aspect van de applicatie code naar eigen inzicht aan te passen;
- sterke Java en J2EE programmeer vaardigheden.

Een argument dat in deze opsomming ontbreekt, is het kostenaspect. De reden hiervoor is dat het kostenaspect afhankelijk van de situatie pleit voor de traditionele productlijn of voor J2EE. Met de huidige stand van zaken kun je stellen dat de deploymentkosten bij het gebruik van de huidige Java tooling en J2EE infrastructuur lager liggen dan bij het gebruik van Designer/Developer (je kunt volstaan met een goedkope J2EE Application Server –desnoods van een concurrent– zonder de anders benodigde Forms runtime component, de zogenaamde Forms Services), maar door de sterk lagere productiviteit van de Java ontwikkeltools lever je die kostenbesparing weer in tijdens de softwareontwikkeling. Afhankelijk van de licentiekosten van de applicatieserver en de omvang van de applicatie (en daarmee samenhangend de ontwikkelinspanning) slaat de kostenweegschaal daarom naar traditioneel of Java uit.

## Architectuur en integratie

Als je de argumenten bekijkt die Oracle aandraagt voor de keuze tussen traditioneel en J2EE, blijkt naast het kostenaspect ook de term 'architectuur' in beide opsommingen te ontbreken. De interesse voor architectuur-denken in Nederland wordt typerend geïllustreerd door het feit dat op het onlangs gehouden congres van Array Publications, Database Systems 2003, een van de vier parallelle tracks volledig was gewijd aan architectuur. Bij het ontwikkelen 'onder architectuur' wordt elke applicatie niet zozeer als programmeeruitdaging gezien, maar wordt getracht de applicatie in een breder perspectief te plaatsen. Rekening houdend met een soort applicatie-infrastructuur waarbinnen de applicatie in gebruik genomen gaat worden, om op deze manier de legacy-van-de-toekomst te voorkomen. Applicaties worden steeds minder gezien als eilandjes van functionaliteit, maar meer als component in een groter geheel met relaties naar en betekenis voor omliggende systemen. De belangrijkste discussies bij klanten over de keuze voor traditioneel of J2EE komen voort uit dit architectuur-denken. Veelal monden deze discussies uit in een battle-of-the-religions (tussen de respectievelijke Java- en Designer/Developer-adepts) en worden argumenten uitvergroot of verkleind om de gewenste keuze te onderbouwen. De stelling die Java-adepten doorgaans poneren, is dat een traditioneel ontwikkelde applicatie een monoliet is, een ondeelbaar geheel, en niet gebaseerd op enig architectuurprincipe. De argumenten daarbij zijn dan dat er geen scheiding is van data, business logica en presentatie. Bovendien zou er geen gelaagde servicestructuur in zijn te ontdekken en zou de applicatie dus geen herbruikbare functionaliteit bevatten.
Dit is echter volstrekt niet waar. Althans, indien de servicestructuur ontbreekt en er geen scheiding van lagen is, dan is dat niet aan de tooling te wijten. Ook met Java kunnen deze architectuurprincipes geschonden worden. Het staat of valt met de bereidheid (en kennis en kunde uiteraard) van de ontwikkelaars om vanuit de genoemde architectuurprincipes te werken. Als we ons een onbevooroordeelde mening willen vormen of met de traditionele productlijn 'onder architectuur' kan worden ontwikkeld, dienen we hiervoor de recentste methodieken, producten, en frameworks te evalueren.

## Service laag

Laten we daarom even stilstaan bij de mogelijkheden van genoemde elementen van de traditionele productlijn. Allereerst is daar de methodiek, Custom Development Method (CDM), die vier jaar geleden al volledig herschreven is met als doel met behulp van Designer en Developer applicaties te ontwikkelen met een volledige scheiding van presentatie-, business logica- en datalaag. Deze scheiding van lagen is zelfs volledig doorgevoerd in de structuur van de handboeken van deze versie van CDM. Ter ondersteuning van een aparte business logica laag is gelijktijdig met de vernieuwde CDM een framework op de markt gebracht. Dit framework bestaat uit een rules-engine die in de database geïnstalleerd dient te worden en een aantal utility's waarmee de volledige lifecycle van business rules, van analyse via technisch ontwerp tot deployment in de rules-engine

***Indien de service-structuur ontbreekt en er geen scheiding van lagen is, dan is dat niet aan de tooling te wijten***

geautomatiseerd wordt ondersteund. In Optimize jaargang 3, nummer 4 (september 2000) heb ik een uitgebreid artikel gewijd aan CDM en het ondersteunende framework, het CDM RuleFrame. Deze publicatie is nog op het Internet terug te vinden[4]. Alle kanttekeningen die ik destijds in het artikel maakte (onder het kopje 'Rozengeur en maneschijn?'), zijn inmiddels geadresseerd door Oracle, en zijn niet meer relevant. Zelfs de performance van de rules-engine is de afgelopen jaren sterk verbeterd, waardoor het afdwingen van de business rules niet alleen in OLTP maar ook in batch-toepassingen performant genoeg gebeurd.

Het bijzondere van de opzet van het CDM RuleFrame is dat de business logica laag als een service laag wordt gegenereerd. Voor elke tabel in de onderliggende datalaag wordt er een zogenaamde Custom API (CAPI) gegenereerd met daarin een aantal standaard services, tijdens de ontwikkeling toegevoegde custom services en business rules. In Listing 1 is een fragment weergegeven van een dergelijke CAPI. Hierin is te zien dat de validatie van business rules in individuele programma-eenheden, functies, is geïmplementeerd. In deze business rules kan weer gebruik gemaakt worden van de aanwezige standaard en custom services. Dit wordt geïllustreerd door de programmacode van business rule 'br_emp004_ent', waarin een aanroep plaatsvindt naar de generieke servicelaag, naar standaard service 'aggregate value'.

```
.
.
.

function br_emp004_ent
( p_dep_id                          in number
, p_job                             in varchar2
, p_id                              in number
) return boolean
is
/*****************************************************************
Purpose      there may not be more than one clerk in each department.
Remarks

Revision History
When         Who
 Revision    What
_____________________________________

22-02-2000  IDEVCOE
 1.0         Using utility HSU_CRBR (revision 6.0.0.9)
*****************************************************************/
   l_rule_ok BOOLEAN := TRUE;
BEGIN

   -- if the :new.job = 'CLERK' then the number of clerks
   -- must = 1 in this department.
   -- Note: exists_row could have been used too, but
   -- aggregate_value is more flexible when rule changes, e.g.
   -- two clerks per department.
   l_rule_ok := (hsd_emp_capi.aggregate_value
                 ( p_aggregate_function => 'COUNT'
                 , p_aggregate_column   => 'id'
                 , p_dep_id              => p_dep_id
                 , p_job                 => 'CLERK'
                 , p_not_id              => p_id
                 ) = 0
                 and
                 p_job = 'CLERK'
                )
                or
                (p_job <> 'CLERK');

   RETURN l_rule_ok;
EXCEPTION
   WHEN OTHERS
   THEN
      qms$errors.unhandled_exception(package_name||'.br_emp004_ent
(f)');
END br_emp004_ent;

function br_emp005_trs
( p_civil_state                     in varchar2
, p_old_civil_state                 in varchar2
) return boolean
is
/*****************************************************************
Purpose      Allowable transitions for Civil State include (S->M, M->D,
M->W, D->M, W->M)
Remarks

Revision History
When         Who
 Revision    What
_____________________________________

22-02-2000  IDEVCOE
 1.0         Using utility HSU_CRBR (revision 6.0.0.9)
*****************************************************************/
   l_rule_ok boolean := true;
begin

   l_rule_ok := ( p_old_civil_state = 'S' and p_civil_state = 'M' ) or
                ( p_old_civil_state = 'M' and p_civil_state = 'D' ) or
                ( p_old_civil_state = 'M' and p_civil_state = 'W' ) or
                ( p_old_civil_state = 'D' and p_civil_state = 'M' ) or
                ( p_old_civil_state = 'W' and p_civil_state = 'M' ) or
                ( p_old_civil_state is null ) or
                ( p_civil_state is null)
                ;

   return l_rule_ok;
exception
   when others
   then
      qms$errors.unhandled_exception(package_name||'.br_emp005_trs
(f)');
end br_emp005_trs;

.
.
.
```

*Listing 1. Fragment van een Custom API van het CDM RuleFrame*

*[4] 'CDM Advantage 2.0. De nieuwe generatie is compleet': http://www.anewlink.nl/publicaties/opt0009.htm*

Hoewel de services in CAPI's zijn geïmplementeerd in PL/SQL, kunnen ze worden gewrapped tot Java Stored Procedures, of zelfs worden gepubliceerd als webservice. Deze webservices kunnen vervolgens worden aangeroepen vanuit bijvoorbeeld .NET en J2EE applicaties. Dergelijke aanroepen zijn volledig transparant. Hoewel er bij webservices 'onder water' wordt gecommuniceerd op basis van de internetstandaarden XML en SOAP (zie afbeelding 1), wordt dat voor de ontwikkelaars volledig aan het oog onttrokken. Ontwikkeltools uit de .NET en J2EE omgeving hebben wizards waarmee het 'stopcontact' (voor de webservice-publicerende partij) en de 'stekker' (voor de consumerende partij) kunnen worden gegenereerd. Wat dat betreft is Oracle JDeveloper voor een Java-tool erg uitgebreid. Het heeft ook wizards waarmee in PL/SQL geprogrammeerde logica als webservice gepubliceerd kan worden. In afbeelding 2a t/m 2d is te zien hoe services van een CAPI met enkele simpele handelingen, zonder enige Java kennis, kunnen worden gepubliceerd. Voor hen die deze activiteit toch liever uitvoeren zonder gebruik van JDeveloper, is er de zogenaamde 'Web Service Assembler'. Dit command-line tool, dat overigens zelf in Java is geschreven, kan op basis van een gespecificeerde configuratiefile (zie Listing 2) de voor het stopcontact benodigde Java-code genereren.

```
<web-service>
<display-name>Web Service using HSD_EMP_CAPI</display-name>
<description>Exposing the PL/SQL package HSD_EMP_CAPI
as a Web Service under the endpoint: /oow/HSD_EMP_CAPI
</description>
<destination-path>./HSD_EMP_CAPI.ear</destination-path>
<temporary-directory>./tmp</temporary-directory>
<context>/oow</context>
<!– A stateless service based on a PL/SQL package. –>
<stateless-stored-procedure-java-service>
<!– The URL under the context /oow for the service –>
<uri>/HSD_EMP_CAPI</uri>
<!– Info needed at publishing time –>
<jar-generation>
  <!– Connection information –>
  <schema>hdemo65/hdemo65</schema>
  <db-url>jdbc:oracle:thin:@localhost:1521:anl33</db-url>
  <!– PL/SQL package information. By default, this is
            also used for the Java class name –>
  <db-pkg-name>HSD_EMP_CAPI</db-pkg-name>
  <!– Java package information –>
  <prefix>anl</prefix>
</jar-generation>

<!– Info needed at runtime - the JNDI DB connection –>
<database-JNDI-name>jdbc/OracleDS</database-JNDI-name>

</stateless-stored-procedure-java-service>
</web-service
```

*Listing 2. Voorbeeld van een Web Service Assembler configuratiefile.*

Zoals we hebben kunnen zien, zijn business rules als zelfstandige eenheden in CAPI's opgenomen. Hierdoor kunnen niet alleen standaard en custom services worden gepubliceerd, maar ook individuele business rules[5]. Dit betekent dat de logica van de business rules ook in andere applicaties kan worden gebruikt, en er dus een zeer geavanceerde vorm van integratie met andere applicaties mogelijk is. Het behoeft geen betoog dat hiermee de investering in de traditioneel geïmplementeerde business logica volledig behouden blijft, ondanks alle technologische ontwikkelingen in de .NET en J2EE hoek. Een schat aan informatie over het publiceren en consumeren van webservices is terug te vinden in de webservices centers van Oracle[6],[7],[8],[9] en Sun[10]. Naast de integratiemogelijkheden van de business logica laag zijn ook de integratiemogelijkheden van de presentatielaag de afgelopen jaren sterk geëvolueerd. Hierdoor kan ook vanuit Developer met Java Componenten worden gecommuniceerd, die als zogenaamde Pluggable Java Components in een Oracle Form kunnen worden opgenomen. In Optimize nummer 1 van dit jaar is hier nog een artikel aan gewijd.
Al met al kunnen we stellen dat de moderne methodieken en productversies in de traditionele productlijn qua architectuurprincipes en integratiemogelijkheden nauwelijks onderdoen voor de J2EE productlijn, temeer daar de services die door de Internet Application Server en de Database Server worden geleverd ook vanuit de traditionele producten kunnen worden aangesproken. De keuze voor traditioneel dan wel J2EE komt dus vooral neer op de eerder opgesomde lijsten met argumenten.

## Aandachtspunten traditioneel

De klanten die hebben besloten gebruik te blijven maken van de traditionele producten dienen wel te beseffen dat ook met deze keuze investeringen gemoeid zijn. We hebben gezien dat het voor moderne, vanuit architectuur gedreven softwaredevelopment een voorwaarde is dat er gebruik gemaakt wordt van de recentere Designer/Developer versies (6i+) en dat het CDM RuleFrame wordt toegepast. Dit betekent dat in ieder geval de productsuites 'Internet Developer Suite' (met daarin Designer/Developer en het dan overbodige JDeveloper) en

*[5] Business rule functions retourneren het datatype BOOLEAN. Omdat dit datatype niet wordt ondersteund door JDBC dient het 'serializable' gemaakt te worden door het BOOLEAN datatype te mappen op een SQL type dat wel wordt herkend door JDBC. Zorg dat script <Oracle_Home>/sqlj/lib/sqljutl.sql in de database is geïnstalleerd.*

*[6] Oracle Webservices Center: http://otn.oracle.com/tech/webservices/*

*[7] Overzichtsdocument Webservice integratie vanuit de database: http://otn.oracle.com/tech/webservices/htdocs/dbwebservices/Database_Web_Services.pdf*

*[8] Publiceren van PL/SQL als webservice: http://otn.oracle.com /tech/webservices/htdocs/series/plsql/content.html*

*[9] Consumeren van Webservices vanuit PL/SQL: http://otn.oracle.com /tech /webservices/htdocs/samples/dbwebservice/DBWebServices_PLSQL.html*

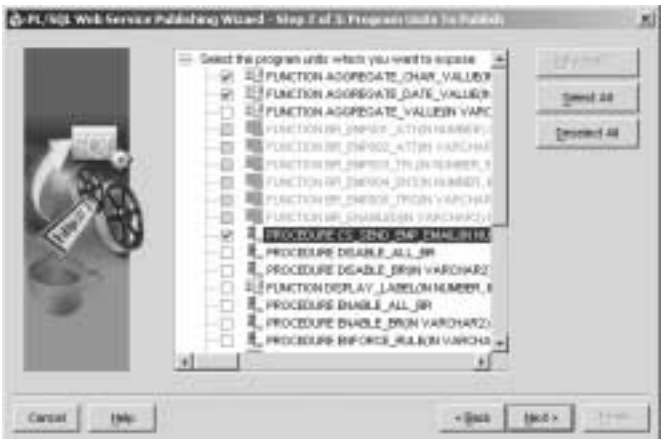
*[10] Sun Webservices Center: http://java.sun.com/webservices*

'iDevelopment Accelerators' (met daarin CDM en Headstart met het CDM RuleFrame en de ondersteunende Utilities) dienen te worden aangeschaft. Deze overstap op nieuwe producten wordt echter niet alleen gestuurd vanuit de drang moderner te ontwikkelen.
De overstap wordt ook actief gestuurd door het supportbeleid van Oracle. Het verstrijken van de supporttermijn van de respectievelijke producten dwingt een upgrade naar een hogere versie af. Dit betekent voor de meest relevante Oracle producten het volgende:

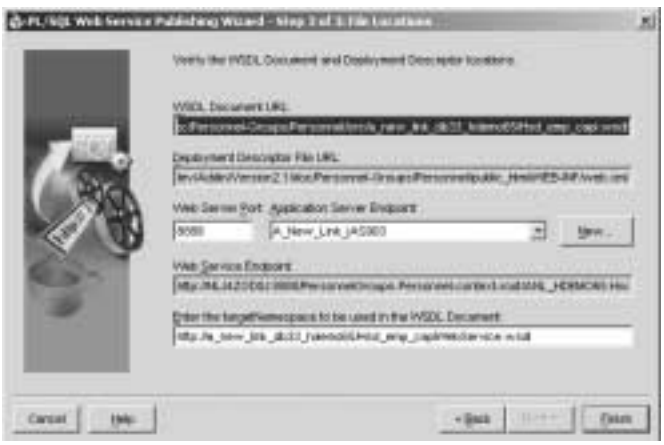
- Oracle Database Server 8.1.7 wordt ondersteund t/m 31 december 2003[11], met de mogelijkheid om dit tegen betaling te verlengen met 2 extra jaren. Een logische opvolger indien nu wordt gemigreerd is Oracle 9i Release 2.
- Oracle Designer 6i wordt ondersteund t/m 31 december 2004[11]. Oracle overweegt momenteel de mogelijkheid te bieden om dit tegen betaling te verlengen met 2 extra jaren afhankelijk van de belangstelling hiervoor.
- Oracle Forms 6i wordt ondersteund t/m 31 december 2004[11]. Oracle overweegt momenteel de mogelijkheid te bieden om dit tegen betaling te verlengen met 3 extra jaren afhankelijk van de belangstelling hiervoor.

Qua Designer en Developer hebben klanten nu de keuze uit twee releases die beide worden ondersteund: 6i en 9i. Een groot verschil tussen deze releases is dat Forms 9i alleen nog maar kan worden gedeployed via het web, dus niet meer in client/server configuratie. Maar wat minstens zo belangrijk is: Oracle heeft release Forms 9i aangegrepen om het Formsproduct te ontdoen van allerlei erfenissen uit het verleden. Functionaliteit waarvan al sinds jaar en dag in de documentatie is aangegeven dat deze 'uitsluitend voor backward-compatibility purposes' in stand is gehouden, en niet zou mogen worden gebruikt, is rücksichtslos verwijderd.
Gedetailleerde informatie over de upgrade is te vinden in het 'Forms Upgrade Center'12. Bij het upgraden van Designer en Developer en het eventueel aanpassen (migreren) van de ontwikkelde programmatuur om van de nieuwe features gebruik te kunnen maken is het belangrijk rekening te houden met de verdwenen backward compatibility van 9i. Zeker als niet direct naar 9i wordt gemigreerd, maar het wel de verwachting is dat dit in de toekomst gaat gebeuren. Het is slecht te verkopen aan de rest van de organisatie als er na een uitgebreide migratie naar 6i niet kan worden volstaan met een eenvoudige upgrade naar 9i. Slimmer is dan om de voor 9i benodigde wijzigingen al in de migratie naar 6i mee te nemen.
Dat door het installeren van nieuwere Designer/Developer versies niet spontaan alle business logica in een separate laag terechtkomt spreekt voor zich. Om van een gescheiden presentatie- , business logica- en datalaag profijt te kunnen hebben zal alle logica gedefinieerd en geïmplementeerd moeten

***Velen hadden het gevoel dat consequente trouw aan één leverancier door Oracle werd misbruikt om haar klanten een compleet andere productlijn op te dringen***



*Afbeelding 2c. Stap 3, het selecteren uit de bron-package van te publiceren program units*



*Afbeelding 2d. Tot slot, het opgeven van deploymentgegevens*

*[11] De gegeven data zijn gebaseerd op door Oracle gepubliceerde informatie ten tijde van schrijven van dit artikel, 29 maart 2003.*
*[12] Forms Upgrade Center: http://otn.oracle.com/products/forms/htdocs/upgrade/content.html*

worden als business rules en moeten worden gedeployed in het CDM RuleFrame. De overgang naar architectuurgedreven ontwikkelen vergt voor applicaties waarin dat nog niet heeft plaatsgehad dus ook een migratie voor de business logica. Door de hoge mate van generatie van deze logica bij toepassing van het CDM RuleFrame wordt de migratie-investering, bij een applicatie waar nog een aanzienlijke hoeveelheid beheer aan deze logica plaatsvindt, vanzelf weer terugverdiend.

## Resumerend

Onder druk van haar klanten is Oracle teruggekomen op haar eerdere strategie. Oracle's standpunt is nu dat zowel de traditionele productlijn met Designer/Developer, als de J2EE productlijn voor haar en haar klanten belangrijk zijn en in de toekomst zal worden voortgezet. Dit betekent dus behoud van investering voor de klanten die gebaseerd op de traditionele productlijn softwareontwikkeling uitvoeren.
Voorts is het een misverstand dat met de traditionele productlijn ontwikkelde applicaties sterk monolithische systemen zouden zijn, als het ware de legacy-van-de-toekomst. Uitgaande van recentere versies in deze productlijn zijn er vele mogelijkheden tot integratie met de buitenwereld via allerlei protocollen. Zo kan bijvoorbeeld via webservices worden geïntegreerd met J2EE en .Net applicaties. Het kiezen voor producten uit de traditionele productlijn betekent niet dat kan worden volstaan met oude niet gesupporte versies. Release 9i komt langzaam in beeld als meest geschikte versie voor Designer en Developer.

***De Statements of Direction doet de uitspraak dat Oracle zowel de traditionele als de J2EE productlijn in volle omvang zal blijven ondersteunen en doorontwikkelen***

Voldoet een organisatie aan de kenmerken voor J2EE ontwikkeling, dan kan worden overwogen over te stappen met alle voordelen van dien. Wel dient beseft te worden, dat deze ontwikkelomgeving op het gebied van Model Based Development in haar kinderschoenen staat, en de productiviteit nog duidelijk achterblijft ten opzichte van ontwikkeling met producten uit de traditionele productlijn.

**Harold Gerritsen**
is Principle Consultant bij A New Link bv. Hij heeft meer dan twaalf jaar ervaring in het adviseren over effectief inzetten van Oracle technologie. http://www.anewlink.nl.
E-mail: h.gerritsen@anewlink.nl

# NIEUWS

## DCS presenteert haar logistieke software op het Oracle 9i platform

DCS Transport and Logistics Solutions, leverancier van software en diensten voor transport en logistiek, maakt bekend dat de DCSi.Logistics software suite per direct leverbaar is op de Oracle9i Database, met een keuze uit de Unix, Linux en Windows NT besturingssystemen. De software die momenteel wereldwijd door meer dan 25.000 eindgebruikers wordt gebruikt, is nu leverbaar op een modern computerplatform dat ruimschoots voldoet aan de huidige eisen met betrekking tot schaalbaarheid en flexibiliteit. DCS Transport and Logistics Solutions breidt haar aanbod uit met de Oracle9i Database om te kunnen voldoen aan de groeiende vraag naar platform-onafhankelijke software in de transport- en logistieke sector. Omdat bedrijven in deze sector steeds vaker transacties met klanten en leveranciers moeten doen via het internet (zoals online orderverwerking, barcode scanning, shipment tracking en elektronische facturering), stijgt de vraag naar software-applicaties die eenvoudig kunnen communiceren met de systemen van derden.
De migratie van de software van haar bestaande IBM DB2 en iSeries (AS/400) platform naar de Oracle9i omgeving is uitgevoerd door een gezamenlijk team van DCS Transport and Logistics Solutions en Oracle. Hierbij is gebruik gemaakt van een speciale migratietool van Oracle partner PKS. De volledige integratie van de DCSi.Logistics Suite en Oracle E-Business Suite moet het in de toekomst bovendien mogelijk maken om een compleet en geavanceerd bedrijfsmanagement systeem voor de transport- en logistieke sector te kunnen bieden. DCSi.Logistics voor Windows NT, Linux and UNIX is per direct beschikbaar. DCS Transport & Logistics Solutions heeft meer dan vijftien jaar gespecialiseerde ervaring in de ontwikkeling en implementatie van systemen bij Transport en 3PL bedrijven. De DCSi.Logistics suite helpt bedrijven om hun belangrijkste bedrijfsprocessen rondom wegtransport, expeditie alsmede third party warehouse management efficiënt en effectief te automatiseren.