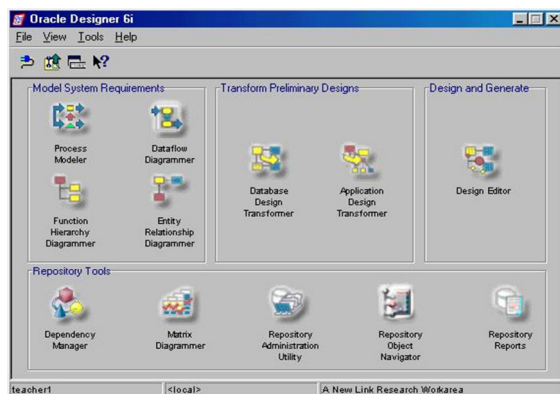


# Oracle Designer 6i

## *Versiebeheer en meer*

Inmiddels hebben we al weer vanaf juli 2000 de beschikking over Oracle Designer 6i. Desondanks kunnen we nog niet echt spreken van een hoge 'adoptiegraad'. Als we kijken hoeveel de nieuwste versie in het veld wordt toegepast, dan zou je gezien de leeftijd van het product anders verwachten. Kennelijk leven er teveel vooroordelen over Designer 6i zoals: te complex, te instabiel en -het versiebeheer uitgezonderd- te weinig nieuwe features. Nu de eerste release van 6i heeft plaatsgemaakt voor z'n opvolger, is het toch echt te kortzichtig het product te blijven negeren. Reden dus om een kijkje te nemen in Oracle Designer 6i.

Als je de gemiddelde ontwikkelaar vraagt wat Designer 6i inhoudt, dan volgt een antwoord in de trant van 'het is eigenlijk Designer 6.0, maar dan met versiebeheer'. Op zich juist, maar het is niet het hele antwoord. Inderdaad hebben nu perfecte mogelijkheden voor versiebeheer, maar om ook eens wat anders te noemen: we kunnen eindelijk Java Classes kwijt in de Repository, we kunnen Oracle Forms-layouts genereren zoals we die willen hebben en we kunnen daarbij gebruik maken van de meest recente features van Oracle Forms 6i. In een tweetal artikelen zullen we aandacht besteden aan deze onderwerpen.

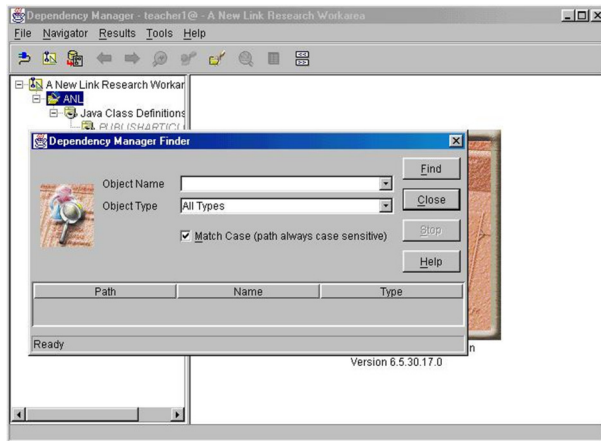


Afbeelding 1 Oracle Designer 6i Frontpanel

Het artikel in deze Optimize zal voornamelijk ingaan op de mogelijkheden van configuratiemanagement en autorisatie binnen de repository, in de volgende uitgave zullen we met name aandacht besteden aan de verbeteringen met betrekking tot het genereren van Oracle Forms. De keuze is hierop gevallen omdat dit de voor Forms-ontwikkelaars belangrijkste verbeteringen zijn, en die vormen de grootste groep Designer gebruikers. Voor de volledigheid dient echter te worden opgemerkt, dat deze onderwerpen slechts een selectie vormen van de nieuwe mogelijkheden van Oracle Designer.

### Oracle Designer 6i

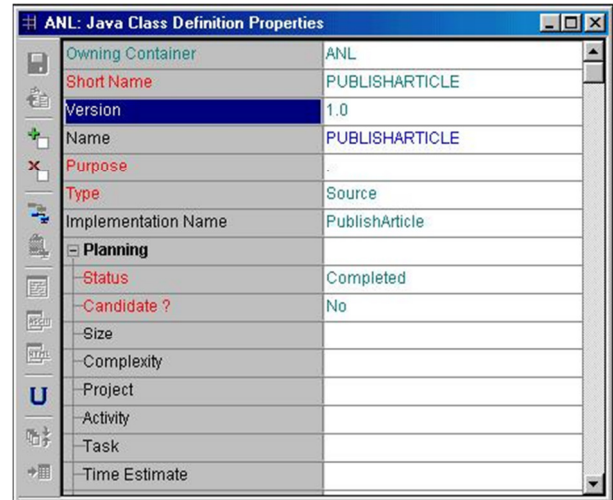
Als we Oracle Designer 6i opstarten zien we het product zoals is weergegeven in afbeelding 1. Vergeleken met Designer 6.0 valt op dat er eigenlijk niet zoveel veranderd is aan de buitenkant. De iconen zijn wat ergonomischer geworden (het ontbreken van het vierkantje om de iconen onderdrukt de neiging die sommigen hebben tot dubbelklikken), en er is slechts één extra tool te ontdekken: de Dependency Manager. Dit tool is, zoals veel nieuwe functionaliteit in Designer 6i, geschreven in Java. Een keerzijde hiervan is dat deze Javautilities veel meer memory gebruiken dan overeenkomstige windowsvarianten. Worden deze utilities dan ook gebruikt, dan is een extra geheugenuitbreiding tot 128 Mb zeker aan te bevelen. Met de Dependency Manager kan perfect een impact analyse van een voorgenomen wijziging worden gedaan. Het geeft aan welke softwareonderdelen afhankelijk zijn van andere. Het fraaie van het tool is dat het in feite een framework is, waarin specifieke 'parsers' kunnen worden gehangen. Hierdoor is de dependency analyse niet beperkt tot objecten in de repository en sourcebestanden van Forms, Reports etc. (deze parsers zijn uiteraard standaard aanwezig), maar kan er voor elk type bestand een parser worden gemaakt en worden opgenomen in de analyzer. Oracle zal dit waarschijnlijk aan third parties over laten. Afbeelding 2 geeft een indruk van de Dependency Manager.



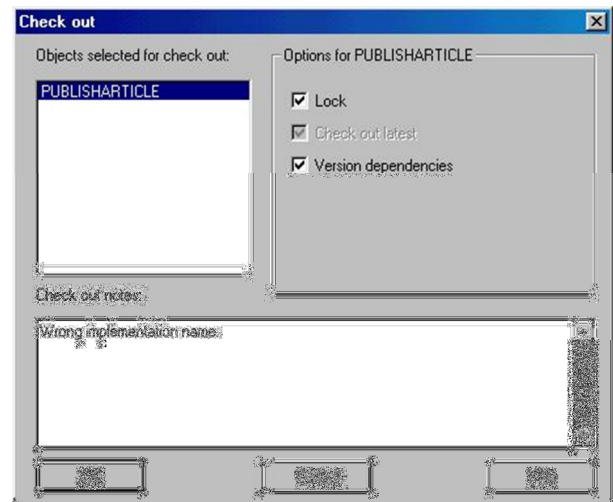
Afbeelding 2 Dependency Manager

## Configuratiemanagement

In relatie tot Oracle Designer is de term versiebeheer eigenlijk niet echt flatteus te noemen. Configuratiemanagement dekt de lading beter. Met Designer 6i kunnen we immers eindelijk de ontwikkeling van de meest omvangrijke en complexe systemen, gebruikmakend van de meest ingewikkelde projectaanpak (bv. incrementeel, iteratief en parallel ontwikkelen), vastleggen in de Repository. Dit moest ook wel, want het beleid van Oracle Corporation schreef jaren geleden al voor dat in de toekomst alle standaardpakketten ('Oracle Applications') gegenereerd zouden moeten worden vanuit Oracle Designer. En aangezien deze pakketten vele duizenden Forms en Reports beslaan, was de vroegere starre structuur van de Repository hiervoor veel te beperkt. Het product dat daarom alle eer toekomt met betrekking tot de mogelijkheden op het gebied van configuratiemanagement is niet zozeer het nieuwe frontend tool, Oracle Designer 6i, maar vooral de onderliggende nieuw ontwikkelde repository, Repository 6.5. Ruim een jaar geleden is in de Optimize al een uitgebreid theoretisch artikel (de producten waren toen nog niet beschikbaar) gepubliceerd over de versiebeheermogelijkheden van de repository. Om niet in herhaling te vervallen, zullen we de belangrijkste aspecten van het product nog eens voor het voetlicht brengen, en -zoals inmiddels mogelijk is- illustreren met screenshots. De volledigheid gebiedt overigens te zeggen dat er met Oracle Designer 6i niet verplicht dient te worden gewerkt met versies. Het is ook mogelijk te blijven werken in de 'non-versioned world'. In dat geval ondersteunt de repository niet meer dan één versie van een bepaalde objectdefinitie. Zodra het versiemechanisme echter wordt aangezet (de optie 'Enable Version Support' in het 'Options'-menu van de Repository Administration Utility) wordt de repositorystructuur fysiek gewijzigd, en is er geen weg meer terug.



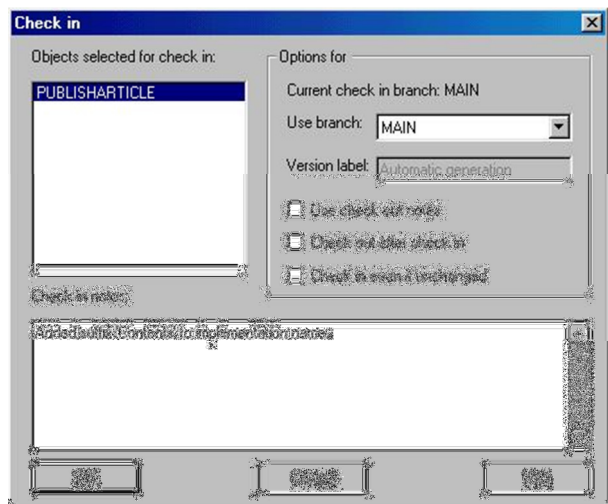
Afbeelding 3a Element vóór check-out



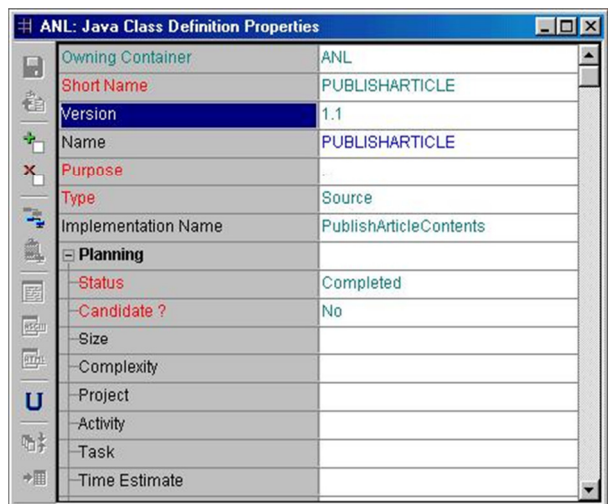
Afbeelding 3b Check-out operatie



Afbeelding 3c Bewerkt element na check-out



Afbeelding 3d Check-in operatie



Afbeelding 3e Element na check-in

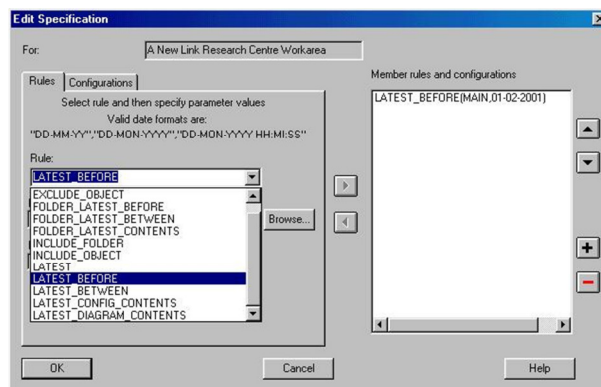
## De versiedimensie

Iedereen liep er in de vorige versies van Designer wel eens tegenaan: terwijl de definitie van een bepaald object in Designer is 'verderontwikkeld', blijkt er toch een probleem te zijn in het reeds uitgeleverde object dat op basis van de 'vorige versie' van de betreffende definitie was gegenereerd. Of nog erger: twee projectteams maken beiden van dezelfde objectdefinitie in Designer gebruik, maar een van de teams wil daar reeds een aanpassing op maken terwijl het andere team op dat moment nog maar toe is aan de 'huidige versie' van het object. Designer kwam je in beide gevallen niet echt tegemoet. In de praktijk probeerde je deze problemen te voorkomen door je werkwijze daarop af te stemmen; door releasegewijs op te leveren c.q. door geen parallelle ontwikkeling van geïntegreerde systemen toe te staan in je project.

Aan deze problemen is nu een einde gekomen. Elk Primary Access Controlled element ('PAC'), zoals bijvoorbeeld tabeldefinities (de bijbehorende kolomdefinities zijn dan de Secondary Access

Controlled elements, 'SAC'), kan na elke wijziging waarvan de ontwikkelaar het van belang vindt dat deze wordt bewaard, worden vastgelegd als een aparte versie van dat element. Zo kan het toevoegen van een kolom aan een tabeldefinitie de tabel een hogere versie doen krijgen. Welke wijzigingen tezamen een nieuwe versie vormen wordt zoals gezegd bepaald door de ontwikkelaar. Alvorens een definitie kan worden aangepast, dient de ontwikkelaar het betreffende object 'uit te checken'. Naar keuze wordt de definitie hierbij gelocked, waardoor het object niet ook (per ongeluk) kan worden uitgecheckt door een andere ontwikkelaar. Het uitgecheckte object kan worden aangepast en getest door de ontwikkelaar (de aanpassingen zijn nog niet zichtbaar voor andere gebruikers van de objectdefinitie) en zodra alle gewenste wijzigingen zijn doorgevoerd, dient de ontwikkelaar het object weer 'in te checken'. Deze stappen worden geïllustreerd in afbeelding 3a t/m 3e. Hierbij kent de repository het object een hoger versienummer toe.

Er is weinig verbeeldingskracht voor nodig om in te zien hoeveel (versies van) objecten er na enige tijd ontstaan in de repository. Het is dan ook bijna ondoenlijk de ontwikkelaars te belasten met het selecteren van de juiste versie van het object dat zij nodig hebben. Sterker nog: alle tools in Oracle Designer zouden nodeloos veel ingewikkelder worden als telkens door de ontwikkelaar een keuze diende te worden gemaakt voor een specifieke versie. Het maken van een ERD in de Entity Relationship Diagrammer bijvoorbeeld, zou voor het in een diagram includen van een entiteit worden ondersteund met een list of values, die niet alleen alle nog niet geïnclose entiteiten bevat, maar ook alle versies daarvan. Om die reden is in Designer (en in de Repository) een concept uitgewerkt dat de versiedimensie voor ontwikkelaars aan het oog onttrekt. Dit concept is geïmplementeerd met behulp van de zogenaamde 'Workarea'.



Afbeelding 4 Specificeren van Rules in de Workarea Wizard

## Workarea's en Configurations

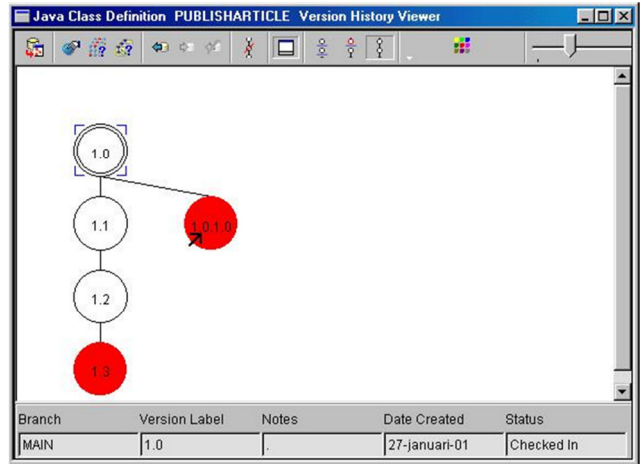
Selecteerde een ontwikkelaar vroeger een Application System als context waarbinnen gewerkt ging worden, tegenwoordig dient voorafgaand daaraan een Workarea geselecteerd te worden. De workarea bepaalt van elk object welke versie daarvan wordt bedoeld. Zodra dus een workarea is gekozen, is er slechts één versie van elk object zichtbaar. De échte complexiteit, nl. het inzicht hebben in alle versies en weten wie welke versie op welk moment te zien moet krijgen, is zodoende verplaatst naar het definiëren van de workarea's. Deze taak, die gezien de impact ervan het best door een zeer ervaren repositorygebruiker kan worden ingevuld, dient apart te worden geautoriseerd (zie verder, Autorisatie). Een ontwikkelaar die deze taak uitvoert (te omschrijven als configuratiebeheerder), zal dit doen gebruikmakend van de Workareawizard. Hiermee kunnen workarea's worden gedefinieerd door het specificeren van de zogenaamde Workarea Rules. In afbeelding 4 wordt deze wizard geïllustreerd met als voorbeeld de rule die specificeert dat van alle objecten de laatste versie voor 01-02-2001 wordt geselecteerd. Zodra een ontwikkelaar een workarea 'opent' worden de gespecificeerde rules van de workarea geëvalueerd en worden de juiste versies zichtbaar. Als er nadien elders een nieuwere versie van een object ontstaat die op grond van de workarea rules zichtbaar zou moeten zijn, dan is dat niet automatisch het geval. Daartoe moet een ontwikkelaar expliciet van de menuoptie 'Refresh Workarea' gebruikmaken. Naast de dynamische workarea's (waarbij het resultaat afhankelijk is van het moment van evalueren van de rules) bestaat er een statischer variant: de Configuration. Een configuration wordt ook gespecificeerd m.b.v. rules, maar deze worden uitsluitend op het moment van definiëren van de configuration geëvalueerd en vertaald naar verwijzingen naar de gespecificeerde versie van de objecten. Een configuration is dan ook niet te refreshen. Door het verschil in aard van de workarea en de configuration hebben ze een verschillend toepassingsgebied: is de eerste vooral bruikbaar tijdens ontwikkeling, de laatste is bij uitstek geschikt om de inhoud van een oplevering te specificeren.

## Versiebeheergereedschappen

Zoals reeds eerder opgemerkt is er in Oracle Designer bewust voor gekozen om de versie-dimensie voor de ontwikkelaars zoveel mogelijk aan het oog te onttrekken. De keerzijde hiervan is dat er via de reguliere tools in Designer (de Diagrammers, de Repository Object Navigator, de Design Editor) niet gemakkelijk inzicht kan worden verkregen in de gegevens die juist wel met de versie-dimensie te maken hebben. Hierbij valt bijvoorbeeld te denken aan het verschil in definitie van twee opeenvolgende versies van een bepaald object. Om die reden heeft Oracle in Designer een aantal ondersteunende tools opgenomen op het gebied van configuratiemanagement. Deze zullen hieronder kort worden beschreven:

- Version History Viewer

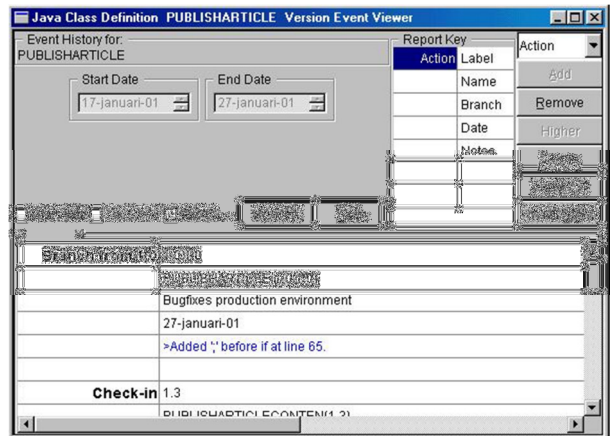
De Version History Viewer is een tool waarmee de gehele versieboom van een object, inclusief eventuele aftakkingen (bijvoorbeeld bij parallele ontwikkeling), grafisch weergegeven kan worden.



Afbeelding 5 Version History Viewer

- Version Event Viewer

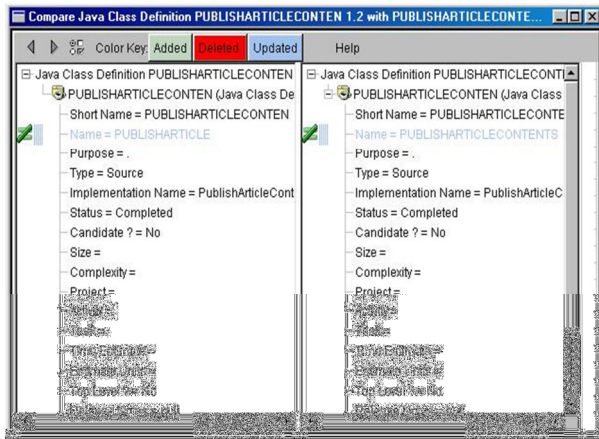
De Version Event Viewer geeft de versiehistorie van een object weer, maar is bedoeld om alle checkin en checkout informatie die door de ontwikkelaar bij een object is vastgelegd, weer te geven.



Afbeelding 6 Version Event Viewer

- Compare Utility

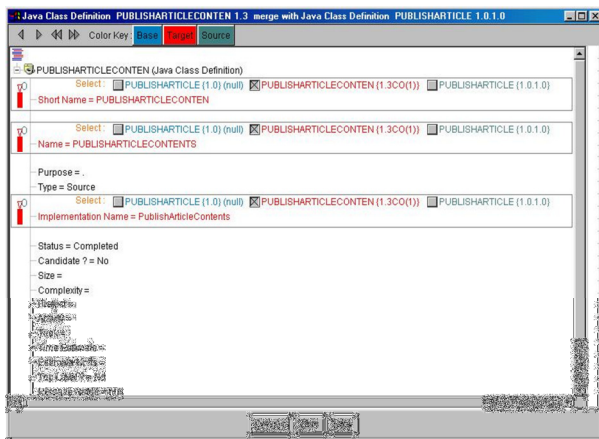
De Compare Utility is vergelijkbaar met de Microsoft utility Windiff. Zoals de Windiff utility op textfiles werkt, zo werkt de Compare Utility op objectdefinities in de repository. Door twee verschillende versies van hetzelfde object te selecteren in dit tool, kan heel snel inzicht worden verkregen in de properties die onderling een afwijkende waarde hebben.



Afbeelding 7 Compare Utility

- Merge utility

De Merge Utility is in feite de Compare Utility met iets verdergaande functionaliteit. Hiermee kunnen twee versies van een object worden samengevoegd tot een derde object. Hierbij kan het samenvoegen automatisch gebeuren volgens een bepaald algoritme, of kan er voor worden gekozen om per property aan te geven of de waarde van de ene versie of die van de andere versie moet worden overgenomen in het nieuw te maken object. Voor ontwikkelaars die deze capriolen in vorige versies van Designer (dus zonder ondersteuning van een utility maar met het betere handwerk) moesten uithalen, is het nauwelijks voor te stellen dat het mergen van versies nu mogelijk is door eenvoudigweg kiezen en klikken. Welnu, het werkt echt!

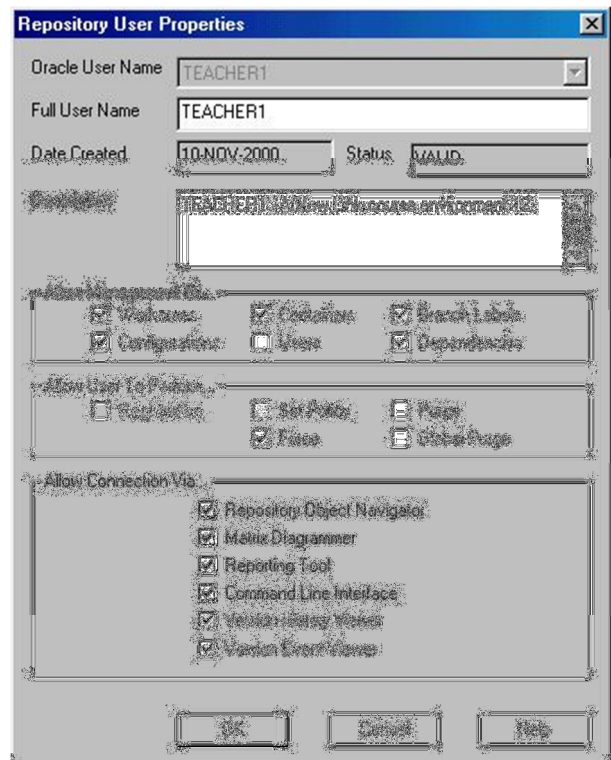


Afbeelding 8 Merge Utility

## Structured en unstructured elements

Dankzij de nieuwe structuur van de onderliggende repository ondersteunt Designer 6i nu ook het opslaan van ('in de ogen' van de repository) unstructured elements. Dat wil zeggen dat naast de bekende Designer-elementen zoals entiteiten, tabellen etc. ook willekeurige Large Binary Objects kunnen worden vastgelegd. Zoals structured elements vroeger onderdeel waren van een

Application System, zo zijn unstructured elements tegenwoordig onderdeel van een zogenaamde 'Folder'. Er is ook een verzamelnaam geïntroduceerd voor Application Systems en Folders: Containers. Hoewel Applications Systems en Folders grotendeels door elkaar kunnen worden gebruikt, adviseert Oracle toch met het oog op toekomstige ontwikkelingen de Application Systems en Folders toe te passen zoals net genoemd (voor structured resp. unstructured elements). Om met unstructured elements te kunnen werken is de menuoptie 'Upload Files en Folders' aangebracht. Zo kan elk bestand (zoals een Microsoft Word document) en zelfs hele directories worden vastgelegd in de repository. Naast upload kan er uiteraard ook gedownload worden, en is er een optie voor het synchroniseren van de inhoud van Designer met dat van het filesystem. Tot slot is er ook een optie om een Folder in Designer te mappen op een plaats in het filesystem. Deze mapping wordt per gebruiker per werkplek vastgelegd zodat gebruik kan worden gemaakt van netwerkdrives. Uiteraard is de achterliggende gedachte van de ondersteuning van unstructured elements dat ook files onder configuratiebeheer van Designer worden gesteld, daarmee de rol overnemend van reguliere producten voor file-versiebeheer zoals PVCS en Clearcase.



Afbeelding 9 User properties in de RAU

## Autorisatie

De afgelopen jaren zijn de mogelijkheden van Oracle Designer op vrijwel alle gebieden heel sterk toegenomen. Een uitzondering hierop vormen de autorisatiemogelijkheden binnen Designer. Eigenlijk was er slechts een zeer grove wijze van autoriseren. Zo was het bijvoorbeeld niet mogelijk het gebruik van specifieke onderdelen van Designer te autoriseren, bijvoorbeeld het draaien van Repository Reports. De enige autorisatie die daar enigszins op leek was dat kon worden vastgelegd dat een repositorygebruiker ofwel Manager was ofwel User. Met de Manager-autorisatie had de gebruiker in de RON de toegang tot het Application menu, met mogelijkheden om een applicatie systeem te verwijderen, er een nieuwe versie van te maken (waarbij alle objecten in het applicatiesysteem werden gekopieerd naar het applicatiesysteem met de hogere versie), of er een archive van te doen of juist een restore.

Gelukkig is in Oracle Designer 6i ook de autorisatie eindelijk volwassen geworden. Als we kijken naar afbeelding 9 dan zien we dat er autorisatie kan worden verleend op een veelvoud aan taken in en onderdelen van Designer. Dit maakt het bijvoorbeeld mogelijk om handen en voeten te geven aan functiescheiding op het gebied van administratie en beheer van de repository. Bij veel organisaties was de DBA tot dusver de aangewezen persoon om repositorybeheer te doen. Hij maakte immers toch al database-backups en hij voerde ook nieuwe users op in de database. De redenering was dat hij dan ook wel een backup kon maken van applicatiesystemen in Designer en nieuwe repositoryusers kon toevoegen. Als we de lijn echter zouden doortrekken in deze administratieve repositorytaken zou de DBA ook de Workarea's moeten gaan definiëren die de objectversies bepalen waar de ontwikkelaars mee werken. Een DBA pur sang is echter niet opgeleid (en ambieert dat waarschijnlijk ook niet) om deze werkzaamheden erbij op te pakken.

Willen we met Oracle Designer 6i functiescheiding goed implementeren dan kun je al gauw denken aan een vijftal functies, elk met z'n eigen taken en verantwoordelijkheden: de repositorybeheerder, de configuratiebeheerder, de DBA, de Application administrator en de ontwikkelaar. De taken die horen bij deze functies kunnen globaal als volgt worden geschetst:

- **Repositorybeheerder**  
De repositorybeheerder voert de repositorygebruikers op en specificeert de verschillende promotielevels (zoals 'ontwikkeling', 'acceptatietest', 'productie') als Workarea's.

- **Configuratiebeheerder**  
De configuratiebeheerder is de persoon die de complexiteit van de versiedimensie weet te doorgronden. Hij specificeert de workarearules en de stelt de configurations samen, refresht workarea's en beheert containers (application systems en folders).
- **DBA**  
De DBA is verantwoordelijk voor installatie, configuratie en onderhoud van databases, zorgt voor optimale performance en beschikbaarheid, en voert de databaseuser-administratie.
- **Application administrator**  
De Application administrator is verantwoordelijk voor de beschikbaarheid van de ontwikkelde applicaties. Daartoe zorgt hij dat de databaseobjecten en programmabestanden worden gesynchroniseerd met de inhoud van de repository (voor het betreffende promotielevel).
- **Ontwikkelaar**  
De ontwikkelaar onderhoudt binnen de context van een container gestructureerde repositoryelementen en files.

## Resumerend

Eindelijk hebben we met Oracle Designer 6i een CASE-tool waarbij we de projectaanpak niet hoeven af te stemmen op de beperkingen van de ontwikkelomgeving. Zelfs de meest dynamische omgeving is, voor wat de configuratieproblematiek betreft, nog beheersbaar dankzij de verregaande mogelijkheden van configuratiemanagement. Eerlijkheid gebiedt wel te zeggen dat het vak van configuratiemanagement niet is weggelegd voor beginnende repositorygebruikers. Het is zeker aan te bevelen, alvorens er een project uitgevoerd gaat worden gebruikmakend van een versioned repository, uitgebreid ervaring op te doen met het kiezen van de juiste Workarea's, Configurations en Containers, kortom met het inrichten van de repository.

Kijk voor meer publicaties op <http://www.anewlink.nl/ict/nl/publicaties/>.

(c) Copyright 2001 A New Link bv