Oracle en MDA

productiviteitsboost

Oracle heeft met Oracle Designer en JDeveloper twee verschillende 'ontwikkellijnen' in huis, die beiden een modelgedreven ontwikkelwijze ondersteunen. Voor Designer zijn geen substantiële ontwikkelingen meer te verwachten, voor JDeveloper echter wel. Vooral in de ondersteuning van de modelgedreven ontwikkelwijze wordt momenteel erg veel energie gestoken.

De combinatie JDeveloper met JHeadstart als additioneel product wordt als opvolger van de modelgedreven ontwikkeltool Designer gezien. In dit eerste deel van een tweedelige reeks artikelen worden de volgende aspecten van de nieuwste ontwikkeltool beschreven:
Kan de JDeveloper/JHeadstart combinatie inderdaad als opvolger van Designer gezien worden en wat is de rol van Model Driven Architecture (in dit artikel verder als MDA aangeduid) hierin?
Wat is de betekenis van de huidige 'ommezwaai', gezien vanuit het perspectief van de Designer-specialist mét modelgedreven ontwikkelervaring.

In dit artikel komen de volgende onderwerpen aan bod:
De positionering van de Oracle JDeveloper ten opzichte van Oracle Designer en de paradigmaverschuiving die een overstap tussen deze ontwikkeltools met zich meebrengt.
Uitleg van het MDA-concept, waarna de betekenis van MDA wordt besproken in relatie met het modelgedreven ontwikkelconcept.
Vervolgens wordt Designer bekeken vanuit het MDA perspectief.
De succesfactoren van modelgedreven ontwikkelen worden aan de orde gesteld aan de hand van de ervaringen met Designer.
De randvoorwaarden die het modelmatige ontwikkelen met Designer mogelijk maken, vormen de afsluiting dit eerste deel.

Het tweede deel van deze artikelreeks gaat in op de mogelijkheden van JDeveloper / JHeadstart om modelgedreven te ontwikkelen. Hierna zullen de in de inleiding genoemde aspecten behandeld worden.

Oracle-ontwikkeltools: de verschuiving

Wordt Oracle Designer-kennis obsolete?
Wanneer we de ontwikkeling van de Oracle onwikkeltools over het laatste decennium volgen, dan ontstaat er voor een Designer software engineer een dilemma: betekent doorgaan met Oracle Designer dat je een doodlopende weg ingeslagen bent? Moet je volledig gaan omscholen naar de Java-technologie om weer up-to-date zijn? Daarbij is een wellicht nog belangrijker vraag: wat gebeurt er met de opgedane kennis van modelgedreven ontwikkelen.
Voorgaande vragen zouden we niet zoveel waarde hoeven te hechten wanneer de verandering van relationeel ontwikkelen naar de object georiënteerde ontwikkelwijze met Java technologie niet een erg grote stap was gebleken. Tevens is er ook sprake van geheel andere tooling waarbij het bij de Java-technologie kenmerkende vakjargon van een geheel andere aard is, dan een Designer/ Forms specialist gewend is.

Systeemontwikkeling met Oracle- tools: van verleden tot heden

Met ADF (Application Development Framework) heeft Oracle een belangrijke stap gezet om J2EE 'conforme' applicaties met JDeveloper te kunnen ontwikkelen. Een interessante vraag is of ADF ook model driven development (MDD) ondersteunt, of dat Oracle hier vanaf is gestapt?
Beschouwen we de huidige stand van zaken bij JDeveloper, dan blijkt dat Oracle nog wel degelijk de modelgedreven ontwikkellijn volgt. Hierbij dient wel aangetekend te worden dat Oracle Consulting Nederland met de ontwikkeling van de additionele tool JHeadstart het toepassen van modelgedreven ontwikkelen pas echt interessant maakt: de productiviteit lijkt nu in de buurt te komen van Designer (in combinatie met de Headstart-templates en productivity boosters).

Alvorens in te gaan op de vraag in hoeverre MDA en MDD binnen de Oracle ontwikkeltools van toepassing zijn, eerst een korte uitleg van wat MDA inhoudt volgens de Object management Group (OMG)[1].

MDA in vogelvlucht

Het raamwerk achter MDA

MDA is een benaderingswijze die stelt dat de ontstane modellen in analyse/ontwerpfases, te allen tijde de basis vormen voor het ontwikkelen van een systeem. Door middel van transformaties wordt uiteindelijk de code gegenereerd. Bij MDA staan de modellen en de transformaties centraal. Op zich zelf is dat niets nieuws: de casetools die in de jaren tachtig ontstonden streefden dat ook na. Het bijzondere aan MDA is echter dat de genoemde benaderingswijze vanuit de volgende gezichtspunten beschreven wordt:

Portabiliteit: vanuit een platformonafhankelijk model (PIM) kan er naar een 'willekeurig' platformspecifiek model (PSM) getransformeerd worden. Hiermee is een platformonafhankelijk model portabel te noemen.

Interoperabiliteit: vanuit een platformonafhankelijk model kunnen verschillende platformafhankelijke modellen getransformeerd worden. Deze dienen ook onderling met elkaar dienen te communiceren. MDA voorziet ook in deze 'PSM naar PSM' brugfunctionaliteit (bijvoorbeeld cross platform interoperabilitity)

Herbruikbaarheid: bij de ontwikkeling van verschillende systemen, blijken de diverse transformatieslagen tussen de modellen, vaak volgens eenzelfde patroon te verlopen. Herbruikbaarheid van deze transformaties is dan ook een groot voordeel.

Hoe de genoemde gezichtspunten gebruikt kunnen worden in een ontwikkelproces is in onderstaande figuur gevisualiseerd en bestaat uit vijf stappen:

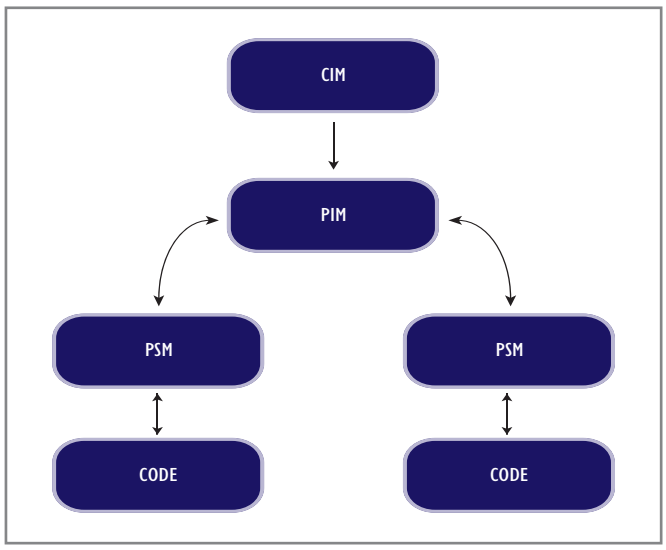
Definieer de (business) requirements in een CIM (Computation Independent Model).

Maak een PIM (Platform Independent model). Dit is de vertaling van de requirements uit het CIM naar een PIM. Alleen de requirements die in het systeem geïmplementeerd dienen te worden, worden in het PIM opgenomen.

Transformeer het PIM naar een of meerdere PSM's (Platform Specific Model). Bij deze transformatie worden platformspecifieke implementatiewijzen toegevoegd. Hierbij dienen keuzes gemaakt te worden uit de verschillende implementatiealternatieven die voor een specifiek platform mogelijk zijn.

De volgende stap is het transformeren van de PSM naar daadwerkelijke software.

De laatste stap is het deployen van het systeem in een specifieke omgeving (niet in bovenstaande figuur opgenomen).



Figuur 1: MDA modellagen

Van de vijf stappen is de eerste stap (het maken van een CIM) binnen de MDA-definitie niet verplicht, de andere wel. Typerend in de stappen zijn de modellen die telkens als uitgangspunt dienen voor een volgende stap en het transformeren van het ene naar het andere model. Hierbij dient vermeld te worden dat binnen MDA de term transformatie alleen bedoeld wordt bij PIM ➔ PSM en PSM ➔ software. Een CIM wordt niet getransformeerd naar een PIM: een CIM kan er in structuur heel anders uitzien dan een PIM.

Varianten van het MDA ontwikkelproces

MDA onderscheidt drie verschillende varianten van het ontwikkelproces: forward engineering, partial engineering en full engineering.

- Forward engineering -

In de basis volgt de ontwikkelcyclus bij MDA hetzelfde patroon als bij de 'traditionele'aanpak. Zie in figuur 2 de doorgetrokken pijlen naar beneden

Eer is echter een verschil. Indien het systeem aangepast dient te worden, worden de aanpassingen altijd vanuit het ontwerpniveau (PIM) of zelfs vanaf het analyseniveau (CIM) gestart. Deze manier van een systeem aanpassing heet forward engineering. Voordeel van deze werkwijze is dat modellen en transformatieslagen altijd actueel zijn en zodoende ook de actuele stand van de code weergeven.

- Partial engineering -

Dit is een variant op forward engineering. Er mag op een lager niveau aangepast worden. De randvoorwaarde geldt dat de getransformeerde producten zelf niet vanuit een hoger niveau aangepast mogen worden. Het betreft hier dan uitbreidingen of een aanpassing die als 'bypass'om het getransformeerde (deel) product gecreëerd dient te worden. Zie de onderbroken pijlen in figuur 2.

- Full engineering -

Dit is reverse engineering vanuit elke onderliggende laag naar de eerst bovenliggende laag. Zie de gestippelde pijlen in figuur 2.

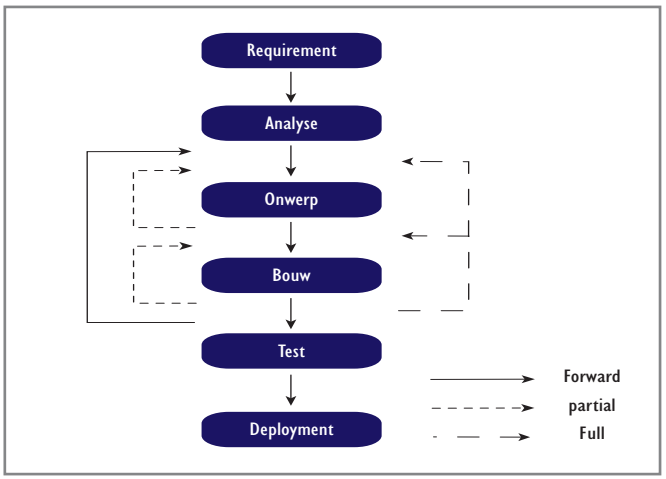
Transformaties
De wijze waarop transformaties plaatsvinden, wordt vastgelegd in een transformatiespecificatie (MDA-mapping). Er zijn diverse manieren om deze transformatiespecificatie vorm te geven. Transformaties kunnen handmatig, door middel van UML profiles, patterns en markings of geheel automatisch gebeuren.

Model Driven Architecture of Model Driven Development?

In diverse artikelen, whitepapers en afstudeerscripties wordt gesproken over MDA enerzijds en MDD (Model Driven Development) anderzijds. Met MDA wordt meer het raamwerk bedoeld, zoals dit door de OMG beschreven wordt en dat op dit moment erg in de belangstelling staat. Sommigen auteurs stellen zelfs dat MDA een nieuwe ontwikkeling in software development is. MDD wordt meer geassocieerd met de casetools, zoals die aan het einde jaren tachtig ontstaan zijn en voorziet in een raamwerk voor een specifiek platform. Platformonafhankelijke modellen (PIM) werden volgens de auteurs niet binnen de toenmalige casetools opgenomen en daarmee ook niet in de bijbehorende tranformatieslagen. In dit artikel wordt MDD niet volgens bovenstaande definitie gebruikt. Model Driven Development staat hier voor de basisgedachte die achter zowel MDA als MDD (de casetools) zit: om vanuit modellen in transformatiestappen naar gegenereerde code te komen.

MDA betekent dan ook niet zo zeer een nieuwe ontwikkelrevolutie, want een tool als Designer in combinatie met Headstart biedt al jaren de mogelijkheid om modelgedreven te ontwikkelen, waarbij 100 procent codegeneratie redelijk wordt benaderd.

MDA, zoals door OMG gedefinieerd, geeft echter het modelgedreven ontwikkelen wel een nieuwe impuls. Dit wordt veroorzaakt door de volgende aspecten:
MDA is gebaseerd op internationale (open) standaards, die sinds de opkomst van de web technologie pas echt op gang is gekomen. Het OMG, die ook een belangrijke rol in deze standaardisatieontwikkeling speelt, kan nu ook gebruik maken van deze standaardisatieontwikkeling. Frappant hierbij is dat toolleveranciers 180 graden gedraaid zijn: hielden de meeste toolleveranciers jaren geleden wel van het 'lock-in' principe vanwege commerciële binding, nu benadrukt vrijwel elke toolleverancier, bang voor uitsluiting van de markt, dat ze conform standaard x, y of z werkt en deze uitwisselbaarheid mogelijk maakt.
Het bovengenoemde aspect biedt nog een ander interessant gegeven: door de toepassing van internationale (open) standaards, leent MDA zich ook uitstekend voor het ontwikkelen van componenten en services in gedistribueerde omgevingen. En dat past nu prima in de webtechnologie.



Figuur 2: Ontwikkelprocesvarianten

Het gebruik van UML als standaard modelleertaal[2] draagt ertoe bij dat er ook op modelleergebied een meer uniforme werkwijze gebruikt wordt en minder afhankelijkheid van tools bestaat. Dit stimuleert de herbruikbaarheid en uitwisselbaarheid.
Opmerkelijk is echter, dat de belangrijkste doelstelling die leveranciers met de modelgedreven ontwikkelwijze nastreven - namelijk productiviteitsverbetering van het ontwikkelproces - niet in de doelstellingen van MDA voorkomen.
Deze productiviteitsverbetering staat in rechtstreeks verband met de mate waarin de transformaties al dan niet geautomatiseerd kunnen verlopen. Dit is weer afhankelijk van de mate waarin transformaties via een aantal vaste patronen kunnen worden uitgevoerd. De stap om vanuit een model de software te genereren is de stap die de meeste snelheid/hoogste productiviteit oplevert.

Nader beschouwd kan wel verklaard worden waarom de OMG met MDA deze doelstelling niet (rechtstreeks) nastreeft: MDA beschrijft modelgedreven ontwikkelen als ontwikkelmethode die breed over alle te ontwikkelen systeemtypen toepasbaar dient te zijn. Het al dan niet geautomatiseerd uitvoeren van de transformaties vormt 'slechts' een onderdeel. Leveranciers zoals Oracle met Oracle Designer behalen een significante productiviteitsverbetering door zich te focussen op een bepaald type systeem en daarop hun transformaties af te stemmen. Hiermee onderscheidt MDA zich ook van de visie, die door Oracle Designer en zoals later zal blijken, door Oracle ADF/ JHeadstart wordt nagestreefd.

Betekenis MDA

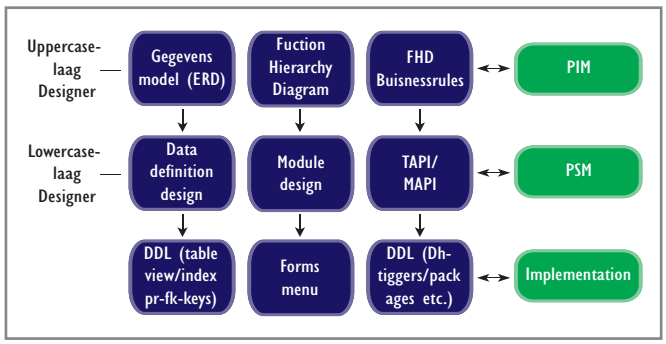
MDA is een architectuur die gebruikt kan worden indien modelgedreven ontwikkelen het te volgen ontwikkelscenario is. In hoeverre een toolleverancier MDA geïmplementeerd heeft, is vrij invulbaar. Dit hangt uiteraard ook af van welke visie en doelstelling de leverancier met de tool heeft. Een MDA-tool

bestaat dan ook niet, wel kan gesteld worden in hoeverre een ontwikkeltool aan MDA voldoet. Deze stelling zegt nog steeds niet hoe werkbaar een tool in de praktijk is. Hiervoor moet toch naar de invulling van de MDA concepten gekeken worden. En dat daar nog een hele wereld achter zit, mag blijken uit de analyse van Oracle's visie op MDA, of liever gezegd: modelgedreven ontwikkelen.

De Oracle ontwikkeltools

Oracle Designer

Oracle Designer is de ontwikkeltool die modelgedreven ontwikkelen ondersteunt en waarbij de basis van het systeem gebaseerd is op Forms. Belangrijke pijlers zijn het ERD (Entity Relational Diagram) en PL/SQL als programmeringstaal. Wanneer de verschillende modellen die in MDA beschreven zijn, afgebeeld worden op de Designer modellen, zijn zeker enkele overeenkomsten te zien:



Figuur 3: MDA-vergelijking met het Oracle Designer concept

De platformonafhankelijkheid van de uppercase-laag in Designer wordt maar gedeeltelijk ondersteund: het FHD (Functional Hierarchy Diagram) en de wijze van vastleggen van business rules zijn zeer sterk gericht op het kunnen transformeren naar het module-design niveau in Designer en zijn daarmee zeer specifiek te noemen voor het Oracle platform. Het ERD kan wel enigszins platformonafhankelijk genoemd worden, zij het dat de ERD modeller in Designer ook zijn eigen standaards heeft.
Het principe van forward engineering is maar ten dele mogelijk: de gegevenszijde van een te ontwikkelen systeem in Designer biedt wel de mogelijkheid tot forward engineering, maar de gedefinieerde functies in de uppercaselaag (analyseniveau) bieden slechts een eenmalige transformatieslag.

Het PSM (lowercase laag) in Designer biedt voor een groot deel van het systeem generatoren om van design models (technische modellen) naar code te genereren. Er dient nog steeds aanvullende code geschreven te worden (t.b.v. business rules), weliswaar beperkt, maar toch essentieel voor het goed functioneren van het systeem.

UML wordt door de OMG gepositioneerd als de standaard te gebruiken modelleertool bij MDA. hierdoor lijkt MDA zich voornamelijk af te spelen in die omgevingen waarbij de webtechologie (lees Java /.Net technologie) de basis vormt voor het PSM.

In die hoedanigheid wordt de al bestaande en bewezen modelgedreven ontwikkeltooling zoals Oracle Designer te kort gedaan. Vele concepten die bij MDA gehanteerd worden (bijvoorbeeld de templates en parametrisering ter ondersteuning van de sturing bij transformaties), worden in Oracle Designer al jaren toegepast.

In dat opzicht is er veel ervaring opgedaan met modelgedreven ontwikkelen en het is zeker interessant deze ervaring eens onder de loep te nemen.

Wat maakt modelgedreven ontwikkelen nu succesvol?

De ervaringen met Oracle Designer in combinatie met Headstart (templates en utilities) stellen ons in staat de volgende constateringen te doen:
Een belangrijk doel van modelgedreven ontwikkelen is het versnellen van het ontwikkelproces. Cruciaal is de mate waarin de transformaties geautomatiseerd zijn. Primair verantwoordelijk voor de versnelling is de configureerbaarheid van de transformaties met behulp van templates en parameters.
Transformatieslagen blijken erg toolafhankelijk opgezet te worden. Dit betekent dat de toolkennis van een modelgedreven ontwikkeltool een andere belangrijke factor voor het behalen van de beoogde productiviteitsverbetering is.
De wijze waarop een engineer de modelgedreven ontwikkeltool volgens de filosofie van de makers gebruikt is ook een belangrijke factor in de productiviteit. De mogelijkheden die de tools bieden zijn zo groot, dat zonder enige begeleiding in deze mogelijkheden, de kans erg reëel is dat een duidelijk minder efficiënte werkwijze gekozen wordt. Goede roadmaps, standaards en richtlijnen zijn onontbeerlijk gebleken.
Modelgedreven ontwikkelen kan als ontwikkelmethodiek zeer goed worden ingezet bij informatiegedreven systemen. Vooral administratief ondersteunende systemen kunnen relatief snel ontwikkeld worden, waarbij meestal de gegevensopslag een belangrijke basis vormt en de functionaliteit die deze gegevens kan manipuleren (via een Mens Machine Interface), niet heel erg complex is[3].
Doordat er snel vanuit een model naar werkende software gegenereerd kan worden, kunnen er zeer snel prototypes gemaakt worden, die in een ontwerpfase gebruikt kunnen worden om de functionaliteit met gebruikers af te stemmen. Het verhoogt de bruikbaarheid van een systeem en de betrokkenheid (acceptatiegraad) van gebruikers bij het ontwikkelen van het systeem.

Modelgedreven ontwikkelen betekent een verschuiving van tijdsbesteding in de verschillende fases waaruit een ontwikkeltraject bestaat: aan de ontwerpfase (lees het modelleren) dient veel aandacht besteed te worden. Immers, deze modellen vormen - samen met het configureren van de transformatie - de basis voor de feitelijke generatie van de code. Zodoende kan er meer aandacht aan de functionaliteit van een systeem besteed worden en bestaat het werk van een ervaren transformatiespecialist (met veel toolkennis!) hoofdzakelijk uit het 'fine-tunen' van de transformatieslag. Het uitvoeren van de transformatieslag zelf is dan een relatief eenvoudig en kortdurende activiteit.

Zoals uit bovenstaande blijkt, levert de mate waarin transformatieslagen bestuurd kunnen worden een belangrijke bijdrage aan het succes van modelgedreven ontwikkelen. Immers, bepaalde ontwerpkeuzes kun je via verschillende patterns oplossen. Omdat het verkrijgen van extra snelheid via modelgedreven ontwikkelen juist ligt in het geautomatiseerd uitvoeren van die transformaties, is het erg belangrijk dat de keuzes die bij een transformatieslag gemaakt kunnen worden, ook beïnvloed kunnen worden.

Concluderend kan gezegd worden dat de keuzemogelijkheden in de transformatieslagen recht evenredig zijn met de inzetbaarheid van de tool als productiviteitsverbeteraar.

Randvoorwaarden

De bovengenoemde constateringen zijn uitsluitend gericht op het modelgedreven ontwikkelen zelf, ofwel gericht op de inhoudelijkheid van het productieproces van een te ontwikkelen systeem. Er zijn echter nog een aantal 'randvoorwaarden':
Het kunnen vastleggen van alle informatie (modellen, tekstuele toelichting/aanvulling, transformaties).
Het met meerdere mensen tegelijkertijd kunnen werken, waarbij dezelfde broninformatie gebruikt wordt.
Rolverdeling en autorisatie bij het gebruik van de gegevens in de repository.
Versiebeheer van alle reeds vastgelegde objecten (modellen, tekst, definities in de vorm van properties etc)

Bij Oracle Designer zijn bovenstaande randvoorwaarden vormgegeven in een repository. Hierin worden alle gegevens, die vanaf de start van het ontwikkeltraject tot en met de implementatie vastgelegd. De repository, die uiteraard in de Oracle-database is geïmplementeerd, vormt daarmee een zeer krachtig middel: gegevens kunnen in allerlei verschijningsvormen getoond worden. Versiebeheer en autorisatie zijn met behulp van deze repository uitstekend te regelen en door middel van aparte functionaliteit kunnen kwaliteitschecks uitgevoerd worden (consistentiechecks, hantering standaards, richtlijnen).

Referenties

- The MDA guide version 1.01, OMG 2003
- MOF core specifications version 2.0, OMG, 2006
- Transforming software development: An MDA roadmap, 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

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 OMG is een internationaal consortium, dat internationale computerspecificatie opstelt en onderhoudt. Voorbeelden hiervan zijn UML, XMI, CORBA en natuurlijk MDA. Zie verder www.OMG.com.*
2. *UML is weliswaar de de-facto modelleertaal, maar binnen de MDA-standaard is iedere modelleertaal die aan MOF (Meta Object Facility) voldoet, toegestaan.*
3. *Complex is in deze context relatief: ook in administratief ondersteunende systemen kunnen ingewikkelde userinterfaces voorkomen. Echter, indien men zich specifiek richt op het manipuleren van gegevens, is deze complexiteit relatief gering t.o.v. bijvoorbeeld de graphics die bij computerspelletjes wordt gebruikt.*