Afstudeerscriptie Informatiekunde:

Demystifying Metadata & Metamodel

Radboud Universiteit Nijmegen

Naam:	**Kim Cratsz**
Studentnummer:	**0419028**
Faculteit:	**Faculteit der Natuurwetenschappen, Wiskunde en Informatica**
Instituut:	**Nijmeegs Instituut voor Informatica en Informatiekunde**
Datum:	**10 Augustus 2006**
Versie:	**Versie 1.8**
Afstudeernummer:	**31IK**

Samenvatting

Metadata en metamodellen zijn de onderwerpen van dit onderzoek en de titel van deze scriptie is: “Demystifying metadata & metamodel”. Demystify betekent dat het mysterie wordt verklaard. In deze scriptie wordt het mysterie rond de termen metadata en metamodellen verklaard. In de IT wereld wordt tegenwoordig veel meer gesproken over metadata en metamodellen, maar de werkelijke betekenis van deze termen is nog lang niet goed begrepen waardoor de termen ook niet goed worden toegepast in de praktijk.

Het gebruik van metadata en metamodellen kan verschillende doelen en voordelen hebben, zoals het vergemakkelijken van interoperabiliteit en integratie van systemen. Met behulp van metamodellen en metadata oplossingen kunnen relevante informatie binnen en buiten organisaties (World Wide Web), makkelijker ontdekt worden en hierdoor beter gebruikt worden door organisaties. De uitwisseling van modellen (metadata) en van metamodellen (meta-metadata) is mogelijk tussen producten die dezelfde meta-metamodellen hebben.

Voor dit onderzoek is er een hoofdvraag opgesteld die als volgt luidt:
“*Hoe kunnen metadata en metamodellen als basis dienen voor het genereren van de gewenste informatie?* ”

Het doel van dit onderzoek is om de termen metadata en metamodellen dichter bij elkaar te brengen zodat de samenhang tussen de termen als basis kunnen dienen voor toekomstige ontwerpen met bijvoorbeeld interoperabiliteit en integratie problemen. Om het mysterie rond de termen uit te kunnen leggen is eerst een definitie gegeven van de termen. Hieronder volgt van metadata en metamodel een definitie.

De definitie van metadata luidt als volgt:*“Metadata zijn de eigenschappen van een gegeven informatie zowel fysiek als elektronisch”.*

De definitie van metamodel luidt als volgt:*“Metamodel is een model gemaakt op een hoger abstractie niveau die de regels en constructies (syntax en semantiek) van het onderliggende model aangeeft”.*

Metadata zijn de karakteristieken van een bepaald object. De structuur van een metadata item kan in de vorm van (O, D) worden aangegeven waarbij O een object is zoals een document en D de data is over het object in kwestie zoals titel en auteur. Voorbeelden van metadata zijn naam van een tabel, type data van een kolom, kleur van een object enzovoorts. Metadata kunnen onderverdeeld worden in beschrijvende, gestructureerde en administratieve metadata. Er zijn verschillende toepassingen die metadata als basis gebruiken voor de technologie. Zo maakt het bestandssysteem van Windows gebruik van metadata om bijvoorbeeld bestanden op te zoeken.

Een metamodel benadrukt het feit dat een metamodel de modelleertaal beschrijft op een hoger abstractie niveau dan de modelleringstaal zelf. Een metamodel is dus ook een model, maar op een hoger abstractie niveau. Het beschrijft de essentiële eigenschappen van de taal die gemodelleerd wordt zoals de syntax en de semantiek. Een modelleertaal kan verschillende metamodellen hebben. De modelleertaal ORM heeft dus niet één specifiek metamodel, maar het kan verschillende metamodellen hebben. De reden hiervoor is dat er verschillende versies bestaan van ORM waardoor er ook verschillende

versies van het metamodel kunnen bestaan. Maar de voorkeur of perceptie van een modelleur kan er ook voor zorgen dat er verschillende metamodellen bestaan. Het metamodel geeft dus aan wat er in het model kan voorkomen. Het model is dus een instantie van het metamodel.

Om metadata en metamodellen dichter bij elkaar te brengen is er gebruik gemaakt van het OMG vier lagen model. Dit vier lagen model geeft vier niveaus aan (M0, M1, M2 en M3) voor het classificeren van gegevens. Dit model heeft een hiërarchische structuur en kan gebruikt worden om aan te geven op welke niveau de data, metadata, model en metamodel zich bevinden.

Op M0 niveau bevindt zich de data. Metadata en model bevinden zich op het M1 niveau en metamodel en meta-metadata bevinden zich op het M2 niveau. Als laatste is het M3 niveau. Hier bevindt zich het meta-metamodel. Geconcludeerd kan worden dat een Mn model een instantie is van een Mn+1 laag model. 1) In een n-laag architectuur model M0, M1 … Mn-1, zijn alle elementen van een Mm laag model een instantie van exact één element van een Mm+1 laag model, en voor alle m<n-1 en 2) elke relatie anders dan de "instantie van" relatie tussen twee elementen X en Y suggereert die level(X)=level(Y). Dit betekent dus dat data een instantie is van een model en metadata. Een model is een instantie van een metamodel en metadata is een instantie van meta-metadata.

Het doel van dit onderzoek is het verklaren en dichter bij elkaar brengen van de termen metadata en metamodel. Om het verband tussen de termen aan te kunnen geven is er gebruik gemaakt van verschillende modellen waar de termen in verwerkt zijn. Het uiteindelijke resultaat van het onderzoek is een overkoepelend model waar de termen data, metadata, model en metamodel in passen. Dit overkoepelende model en de andere modellen die gebruikt zijn om de relatie tussen de termen aan te geven zijn te vinden in hoofdstuk 6 (figuren; 6-3, 6-7, 6-9, 6-10 en 6-11).

De conclusie van dit onderzoek is dat de termen metadata en metamodellen uitgelegd zijn door onder anderen een definitie en praktische voorbeelden te geven. Er is een relatie gevonden tussen de termen en deze is verwerkt in een aantal modellen in hoofdstuk 6.

Voorwoord

Voor u ligt mijn scriptie waarin verslag is gedaan van het onderzoek betreffende metadata en metamodellen. Deze scriptie is het resultaat van mijn afstudeeronderzoek aan de Radboud Universiteit Nijmegen. De titel van deze scriptie is: “Demystifying metadata & metamodel”.

Graag wil ik als eerste mijn afstudeerbegeleider Dr. Patrick van Bommel hartelijk bedanken voor de goede begeleiding en tips gedurende de afgelopen maanden. Ik wil iedereen bedanken die mij geïnspireerd hebben tot het volgen van de opleiding Informatiekunde na het afronden van mijn HBO opleiding. Verder wil ik mijn familie bedanken voor hun oneindige steun.

Kim S.M. Cratsz
Arnhem, 10 Augustus 2006

Inhoudsopgave

1. Probleemstelling

In dit hoofdstuk zullen de probleemstelling (waartoe), doelstelling (waarom) en de vraagstelling (wat) van deze scriptie aan bod komen.

1.1 Achtergrond

Gedurende de IT evolutie was de focus gezet op software, want de software kon informatie generen die van belang is voor de organisatie. Gedurende deze periode was de programmeur niet verantwoordelijk voor de interpretatie van de beschikbare informatie, maar alleen verantwoordelijk voor het maken van de software. De informatie was voor gebruik en analyse door de eindgebruiker. IT was toen ook niet verantwoordelijk voor onjuiste of ongeschikte besluiten die gebaseerd waren op gegenereerde rapporten, tenzij de informatie incorrect was. En incorrecte informatie was vroeger het gevolg van een bug in het programma.

Een overeenkomst tussen vroeger en nu is dat we nog steeds in een race zitten van informatie missers [15]. Vroeger konden we de juiste informatie niet genereren en nu kunnen we het nog steeds niet. En om het nog erger te maken, kampen organisaties tegenwoordig met een overbelasting aan informatie. Deze groei heeft te maken met; toename in communicaties zowel intern als extern van de organisatie, globalisering (vrijemarkt), meer bedrijven en dus ook meer competitie, outsourcing enzovoorts. Ook zijn er nieuwe communicatie mogelijkheden zoals fax, telefoon, mail en Internet bijgekomen. Belangrijk om te weten is dat de informatie een onzichtbare goudmijn is voor organisaties en deze goudmijn is nog niet geëxploiteerd [32].

We leven in een tijd waar de technologie niet stil staat, maar technologieën voor het managen van de informatie is vaak het probleem en niet de oplossing. Waaraan ligt dit precies? We zijn in staat om wolkenkrabbers van meer dan 500 meter hoog en vliegtuigen te maken, maar toch kunnen we niet eens zo iets simpels als informatie managen. De beschikbare informatie wordt niet ten volle benut. Goede informatie generen en omzetten tot kennis is niet een eenvoudige taak en het informatieaanbod zal de komende jaren alleen maar blijven toenemen.

Het probleem uit zich in verschillende vormen en op verschillende niveaus. Het genereren van de gewenste informatie moet niet alleen gezien worden als klant gegevens die bekeken kunnen worden met gebruik van een applicatie, want het probleem van de informatie is veel groter dan dit. Het probleem is herkenbaar bij onder anderen de volgende terreinen; Internet (zoekmachines), databases (SQL, RDBS, FDBS), applicaties, bestandssystemen, Image en muziek bestanden, data warehouse en nog veel meer.

Met behulp van modelleringstechnieken wordt er geprobeerd om de processen en informatiestromen in organisaties beter in kaart te brengen om op deze manier software applicaties te ontwikkelen die geschikt zijn voor het genereren van de gewenste informatie. In de literatuur wordt er veel aandacht besteed aan modelleertalen en modelleertechnieken, maar hoe deze talen en technieken samen gebruikt kunnen worden om het probleem op te lossen is niet te vinden in de literatuur. Vooral nu er een teveel is aan technieken en talen is het nodig om een oplossing te vinden. Om de gewenste informatie te kunnen genereren zijn de data, de modelleertaal en de techniek van belang.

Deze drie moeten niet afzonderlijk van elkaar bekeken worden, maar samen in de vorm van metadata en metamodellen.

Tegenwoordig worden de termen metadata en metamodel vaak gebruikt in de IT wereld. Het is bijna een hype geworden, maar deze termen zijn niet nieuw. Metadata hebben altijd bestaan, maar in 1970 kreeg het meer aandacht onder de naam catalog toen Dr. Edgar. F. Codd een theorie startte over relationele databases. Metamodel is een term die vanaf 1980 al gebruikt wordt, maar het is met de komst van Internet, business integratie en de groei aan informatie dat de metamodellen een prioriteit hebben gekregen. Metamodellen en metadata zijn samen de fundamenten voor bijvoorbeeld data integratie. Maar vaak worden deze twee termen verkeerd gebruikt of verkeerd geïnterpreteerd. Hierdoor weten bijvoorbeeld organisaties niet wat de voordelen zijn die met deze termen behaald kunnen worden en blijft de race voor informatie missers bestaan. Metadata en metamodel worden in de literatuur afzonderlijk van elkaar behandeld. Het lijkt bijna alsof het twee dingen zijn die niets met elkaar te maken hebben. Dit is echter niet het geval.

1.2 Doelstelling

Door gebruik te maken van metadata en metamodellen kan de interoperabiliteit en integratie van (legacy) systemen vergemakkelijkt worden. Informatie is heel erg belangrijk binnen organisaties. Zonder informatie (gegevens) kan een organisatie niet bestaan. Met behulp van metamodellen en metadata oplossingen kunnen relevante informatie binnen en buiten organisaties (World Wide Web), makkelijker ontdekt worden en hierdoor beter gebruikt worden door organisaties. Een voorbeeld is om een betere marktpositie te creëren op hun concurrentie. Door gebruik te maken van metamodellen is integratie en interoperabiliteit gebaseerd op metadata mogelijk. De uitwisseling van modellen (metadata) en van metamodellen (meta-metadata) is mogelijk tussen producten die dezelfde meta-metamodellen hebben.

Het doel van dit onderzoek is om de termen metadata en metamodellen dichter bij elkaar te brengen zodat de samenhang tussen de termen als basis kan dienen voor toekomstige ontwerpen met bijvoorbeeld interoperabiliteit en integratie problemen.

1.2.1 Probleemdefinitie en Vraagstelling

Het is bekend dat organisaties de beschikbare informatie niet ten volle benutten en het genereren van goede informatie is niet een eenvoudige taak. Hoe kunnen metadata en metamodellen hierbij helpen? In de literatuur wordt van beide termen wel informatie en onderzoek te vinden, maar vaak worden deze termen verkeerd geïnterpreteerd en ook afzonderlijk van elkaar behandeld. Het is nog niet duidelijk wat men hiermee kan doen.

Hoe verwart zijn de termen data, informatie, metadata, modellen, metamodellen en kennis? Weten gebruikers, administrators, modelleurs, managers enzovoorts wat deze termen precies betekenen en hoe ze met elkaar in verband staan? Het is hierdoor belangrijk om te weten wat de voordelen zijn die behaald kunnen worden door ***goed*** gebruik te maken / toepassen van deze termen. Maar om deze termen goed toe te kunnen passen is het eerst nog belangrijker om te weten wat deze termen zijn.

Voordat er begonnen kan worden aan het bedenken van oplossingen met behulp van metadata en metamodellen is het belangrijk om eerst te weten wat ze zijn. Anders blijft de technologie voor het managen van informatie het probleem en niet de oplossing.

De centrale vraagstelling voor dit onderzoek luidt: "*Hoe kunnen metadata en metamodellen als basis dienen voor het genereren van de gewenste informatie?* "

Zoals aangegeven is het genereren van de juiste informatie een heel breed onderwerp. Het dekt ook zaken als informatie bronnen op het Internet, informatie uit databases, en nog veel meer. In deze scriptie zal onderzocht worden hoe metadata en metamodellen kunnen helpen bij het probleem van de informatie. Het resultaat van dit onderzoek is een overkoepelend model waar de samenhang tussen deze termen (data, metadata en metamodellen) zal worden weergegeven. Het doel van dit onderzoek is om de termen metadata en metamodellen dichter bij elkaar te brengen.

1.2.2 Onderzoeksvragen

Voor de beantwoording van de centrale vraagstelling in deze scriptie is het nodig om de hoofdvraag in kleine onderzoeksvragen onder te verdelen. Hieronder volgen de onderzoeksvragen:

- Wat zijn data, metadata, model en metamodel?;
- Wat is een mogelijke definitie voor de termen metadata en metamodel?;
- Hoe worden deze termen momenteel toegepast in de praktijk?;
- Bestaat er een relatie tussen deze termen?;
- Wat is de relatie tussen deze termen?
- Hoe kan het verband, de relatie tussen deze termen worden weergegeven?

1.3 Afbakening

Deze scriptie richt zich alleen op de beantwoording van de hoofdvraag: "*Hoe kunnen metadata en metamodellen als basis dienen voor het genereren van de gewenste informatie?* ". Om de vraag te beantwoorden zal er een definitie, een betekenis en de relatie tussen deze termen gegeven worden. De toepassingen van metadata en metamodellen in de scriptie zijn voorbeelden. Er zullen geen specifieke of technische oplossingen gegeven worden. De voorbeelden in de scriptie worden gebruikt om de termen uit te kunnen leggen en zijn dus geen handleiding voor implementaties in organisaties. Met andere woorden metamodel en metadata oplossingen voor organisaties zullen niet aan bod komen in deze scriptie.

Er zal in deze scriptie geen specifieke oplossing gegeven worden voor het kunnen genereren van de juiste informatie. Het gaat in deze scriptie om het onderzoeken wat metadata en metamodellen zijn, de onderlinge relaties en hoe ze kunnen bijdragen bij het uitvinden van nieuwe oplossingen.

1.4 Aanpak onderzoekstraject

Het doel van dit onderzoek is om de termen data, metadata, model en metamodel dichter bij elkaar te brengen in de vorm van een overkoepelend model. Om dit te kunnen bereiken zal er gebruik worden gemaakt van het vier lagen model van OMG. Dit vier lagen model dient als basis voor het aangeven van de samenhang. Als eerste zullen de termen afzonderlijk van elkaar behandeld worden, zodat de inhoudelijke betekenis begrepen kan worden. Dit zal als volgt plaats vinden; als eerste zal er van elke term een algemene inleiding gegeven worden met een definitie. Als tweede zal er van elke term toepassingsvoorbeelden gegeven worden die momenteel in de praktijk gebruikt worden. Op technisch niveau zullen er ook formele onderbouwingen plaats vinden. Vervolgens zullen de termen met behulp van het vier lagen model bij elkaar gebracht worden in één groot model, het overkoepelende model.

1.5 Opbouw rapport

Hoofdstuk 1 is de inleiding met de probleemstelling en doel van het onderwerp in deze scriptie. Metadata en metamodellen zullen gebruikt worden om de hoofdvraag in deze scriptie te beantwoorden. Om dit te kunnen doen zal er eerst in hoofdstuk 2 een algemene inleiding van deze twee termen volgen. Vervolgens zal er in hoofdstuk 3 de term metadata verder uitgelegd worden met voorbeelden uit de praktijk. In hoofdstuk 4 komen de systeem tabellen aan de orde. Dit hoofdstuk richt zich meer op metadata in databases. Er is gekozen om dit onderwerp in een apart hoofdstuk uitgebreid ui te leggen, omdat tegenwoordig alle applicaties op databases draaien. Het is dan ook op dit terrein dat er veel aandacht nodig is met betrekking tot oplossingen voor het genereren van de gewenste informatie. Tevens dient dit hoofdstuk ook om de term metadata beter uit te kunnen leggen. In hoofdstuk 5 zal vervolgens de term metamodel aan bod komen. Ook hier zal de term aan de hand van voorbeelden uitgelegd worden en ook een definitie van de term zal volgen. De samenhang tussen de twee termen metadata en metamodel zal in hoofdstuk 6 aan bod komen. Alle onderwerpen die in de voorgaande hoofdstukken besproken zijn, zullen in dit hoofdstuk bij elkaar gebracht worden. Als laatste zal er in hoofdstuk 7 een conclusie gegeven worden met betrekking tot alle voorgaande hoofdstukken, de vraagstelling en een persoonlijke evaluatie van het traject van dit onderzoek.

2. Metadata en metamodellen algemeen

De termen metamodel en metadata worden vaak door elkaar gehaald of verkeerd gebruikt. In de volgende paragrafen zal er een algemene inleiding worden gegeven van deze twee termen.

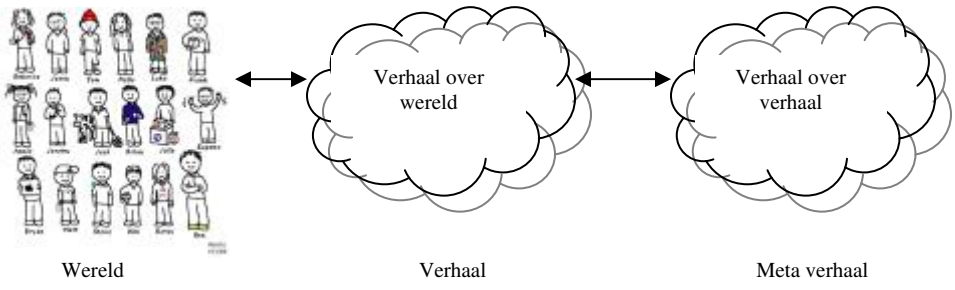
2.1 Meta

Zoals aangegeven worden de termen metadata en metamodel vaak verkeerd gebruikt. Om deze twee termen te kunnen begrijpen wordt er eerst een korte beschrijving gegeven van de woorden model, data en meta.

2.1.1 Model, data en meta

Een model is een voorbeeld van de werkelijkheid. Dit voorbeeld kan ook een systeem worden genoemd dat de wekelijkheid probeert te beschrijven of na te bootsen [42]. Met data wordt bedoeld gegevens, informatie die beschikbaar is zoals adresgegevens. Beide woorden data en model hebben veel met elkaar te maken. Een model beschrijft namelijk de werkelijkheid die weer uit gegevens (data) bestaat en van de beschikbare data kan op zijn beurt weer een model worden gemaakt.

Het woord meta komt uit het Grieks en betekent letterlijk “over”, “verder”. In het Nederlands, maar ook andere talen zoals Engels, wordt meta als een voorvoegsel gebruikt om een concept aan te duiden dat een abstractie is van een ander concept. Metataal verwijst bijvoorbeeld naar een type taal of systeem dat de taal beschrijft [42]. Meta wordt bij Van Dale (taalweb) omschreven als “handelend over het in grondwoord genoemde”. De betekenis van meta hangt echter af van de context waarin het gebruikt wordt. Een meta tag is bijvoorbeeld een HTML tag dat informatie geeft *over* een web document [meer informatie hierover in paragraaf 3.4.2]. Metadata betekent dus ook *simpel* gezegd data over data en metamodel een model over of van een model.



Figuur 2-1 Meta verhaal

De betekenis van meta wordt aan de hand van Figuur 2-1 uitgelegd. De wereld kan gemodelleerd worden in een verhaal. Dit verhaal vertelt iets over de wereld. Het verhaal over de wereld kan op zijn beurt weer gemodelleerd worden in een ander verhaal, een meta verhaal. Met andere woorden het meta verhaal vertelt iets over het eerste verhaal en

het eerste verhaal vertelt iets over de wereld. Bij dit figuur is de wereld een instantie van het eerste verhaal en is het eerste verhaal ook een instantie, maar van het metaverhaal [10].

Tussen metamodel en metadata zijn er enkele verschillen te onderkennen, maar om deze te kunnen begrijpen zullen beide termen eerst uitgebreid worden uitgelegd. In de volgende hoofdstukken zal er worden ingegaan op de betekenis en toepassingen van metadata en metamodel.

3. Metadata

Metadata worden vaak omschreven als “data about data” en de term wordt vaak verschillend gebruikt in de verschillende gemeenschappen. Soms wordt de term gebruikt om te verwijzen naar begrijpelijke machine informatie, terwijl anderen het alleen gebruiken voor records die elektronische bronnen beschrijven [3].

Er bestaat geen officiële definitie van metadata, zelfs de term wordt verschillend opgeschreven; “meta-data”, “metadata” of “meta data”, (in deze scriptie wordt ervoor gekozen om “metadata” aan te houden). Doordat er geen officiële definitie bestaat voor de term, hebben verschillende auteurs getracht om een eigen omschrijving te geven van de term. Dit heeft ertoe geleid dat er tegenwoordig verschillende omschrijvingen bestaat. Hieronder volgen enkele definities van metadata:

- Metadata beschrijven alles wat nodig is om data te kunnen gebruiken. Metadata kan van alles zijn: documenten, databases met schema's en inhoud, programma code, business modellen enz [1].

- Metadata: de gedetailleerde beschrijving van een instance data; het formaat en de karakteristieken van de gepopuleerde instance data; instanties en waarden afhankelijk van de rol van de metadata ontvanger [31].

- Metadata kan gedefinieerd worden als "de informatie die gebruikt wordt om de context en de structuur van data en informatieprocessen te beschrijven" [1].

- Gestructureerde data over data (structured data about data) [42].

- Het begrip metadata omvat voor instellingen van cultureel erfgoed gestructureerde gegevens over het historische materiaal dat wordt bewaard in een instelling (bijvoorbeeld overzichten, catalogi, inventarissen en genealogische indexen), ongestructureerde gegevens over dat materiaal (bijvoorbeeld publicaties en handleidingen) en kennis die bij de medewerkers van de instelling aanwezig is over dat materiaal. De laatste categorie kunnen ook omschreven worden als mobiele metadata, met alle risico’s van dien (ziekte, pensioen, vakantie) [29].

- Metadata is beschrijvende informatie over een object of bron, zowel fysiek als elektronisch [28].

- Metadata zijn de eigenschappen van een gegeven informatie [4].

Metadata betekenen letterlijk (gestructureerde) data over data, maar deze omschrijving is te simpel en nietszeggend. Voor deze scriptie wordt er hierdoor gekozen voor een combinatie van de laatste twee bovenstaande definities; ***“Metadata zijn de eigenschappen van een gegeven informatie zowel fysiek als elektronisch”***. Deze definitie is algemeen, maar het geeft aan wat metadata zijn en kan juist doordat het algemeen is, toegepast worden bij de verschillende disciplines. Wat betekent de definitie precies? Wat is een gegeven informatie en wat zijn de eigenschappen ervan? Een gegeven informatie is in dit geval een object en de metadata zijn de eigenschappen van het object.

Metadata zijn data die de karakteristieken van bepaalde gegevens / objecten beschrijven zoals de naam van een database, tabel, type data van een kolom of toegangsrechten.

Dus de structuur van een metadata item kan in de vorm van (O, D) worden aangegeven waarbij O een object is zoals een document en D de data is over het object in kwestie zoals titel en auteur. Sommige gegevens in D zijn afleidbaar uit O, bijvoorbeeld lengte van het object O. Andere gegevens in D zijn niet afleidbaar uit O, zoals de prijs.

Een metadata item (MI) wordt als volgt weergegeven:

MI= (O,D) waarbij $\begin{cases} \text{O is object} \\ \text{D bevat de data over O} \end{cases}$

Een abstracte specificatie van metadata kan als volgt worden aangegeven:

M= (U,F,I) waarin:

U: U is de onderliggende verzameling van objecten (universum), het domein. Dit kan genoteerd worden als $O \in U$.

F: F is een verzameling van functies over U. Elke $f \in F$ berekent een bepaalde eigenschap van objecten. Bijvoorbeeld een $f \in F$ kan zijn; Bereken de lengte van O. Andere functies die een bepaalde eigenschap van een object kunnen berekenen zijn: bereken aantal pagina's van O of geef samenvatting van O en nog veel meer. Dus F={lengte, #Pagina's, Samenvatting,…}. Dus als $f \in F$ en $O \in U$ dan bevat f(O) de waarde voor de betreffende eigenschap van het object O. Bijvoorbeeld: #pagina's(O).

I: I is een verzameling van metadata items. Er is eerder al aangegeven dat een metadata item als (O,D) kan worden aangegeven, dus stel $(O,D) \in I$. Dan is $O \in U$ een object in het onderliggende universum en D is een specificatie van de eigenschappen van O. Sommige eigenschappen in D zijn niet afleidbaar uit O, bijvoorbeeld de prijs. Andere eigenschappen in D zijn wel afleidbaar met behulp van een functie uit F.

Dus er kan geconcludeerd worden dat voor alle objecten O uit het domein U, er tenminste één eigenschap D uit I bestaat, die een eigenschap is van O.

$\forall O \in U \; \exists D \in I$ (MI(O,D)

3.1 Soorten metadata

Metadata kunnen onderverdeeld worden in drie soorten metadata. Hieronder volgt een beschrijving van deze drie soorten [17] (metadata soort 1):

- ***Beschrijvende metadata***: ·Hiermee kan een bepaalde bron beschreven worden voor bijvoorbeeld ontdekking (zoeken) en identificatie. Het kan elementen bevatten zoals; titel, samenvatting, auteur en andere keywords.
  Beschrijvende metadata kan onderverdeeld worden in contextueel- en semantiek metadata. Met contextuele metadata wordt verwezen naar alle contextuele eigenschappen van een object, voorbeelden hiervan zijn: datum, naam, titel etc. Semantiek beschrijvende metadata zijn bijvoorbeeld geschreven samenvattingen, annotaties, tekst indexen, statistische analyses etc [4];

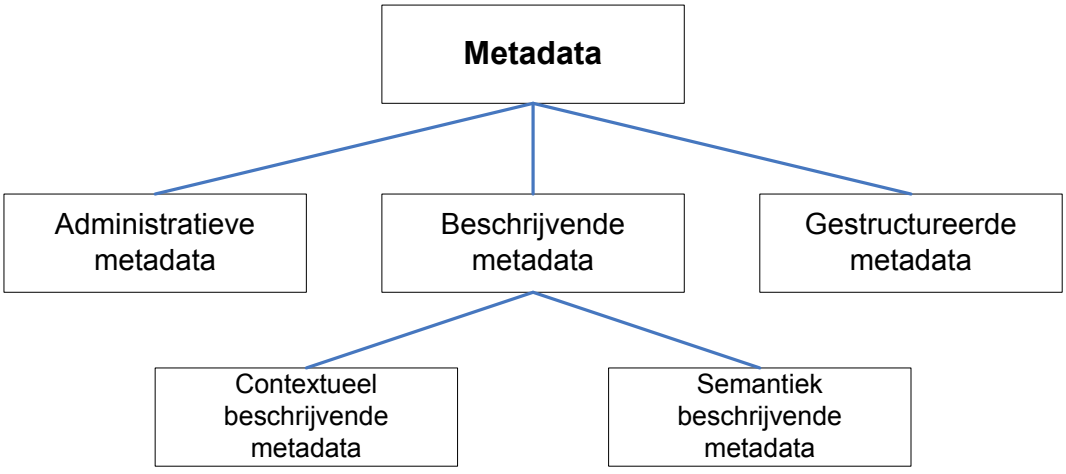
- ***Gestructureerde metadata***: ·Geeft aan hoe objecten kunnen worden samengesteld tot een geheel. Bijvoorbeeld hoe pagina's tot hoofdstukken kunnen worden geordend;

- ***Administratieve metadata***: ·Geeft informatie om een bron te kunnen managen, zoals wanneer en hoe het was gemaakt, bestand type, andere technische informatie en ook wie toegang / rechten heeft tot de bron.

Metadata worden soms ook onderverdeeld in (metadata soort 2) [7]:

- ***Content:*** ·Titel, omschrijving, onderwerp en coverage
- ***Intellectueel eigendom:*** ·Contributor, maker, uitgeverij en rechten
- ***Aanmaak versie:*** ·Datum, formaat, identifier, taal.

De onderverdeling bij metadata soort 2, wordt niet vaak gebruikt in de literatuur. Hierdoor is er gekozen om de onderverdeling van metadata soort 1 in deze scriptie te hanteren. In Figuur 3-1 wordt de samenhang tussen de verschillende soorten metadata weergegeven.



Figuur 3-1 onderverdeling metadata

Stel 'S' bevat de soorten metadata zoals weergegeven in Figuur 3-1.
Dus S = {beschrijvend, gestructureerd, administratief}

Als E alle eigenschappen bevat die op een bepaald moment beschikbaar zijn, dan is E de vereniging van alle eigenschappen D die in I voorkomen.

E = de vereniging van alle D die in I voorkomen.

Om van elke eigenschap vast te leggen welke soort metadata het betreft, hebben we dus een functie 'Soort' nodig voor de volgende formule:

Soort : $E \rightarrow S$

Dus de functie Soort bepaalt tot welke soort metadata, de eigenschap D behoort.

Bijvoorbeeld voor een bepaalde eigenschap $x \leftarrow E$ kunnen we hebben
Soort(x) = administratief.

3.2 Metadata begunstigden / belanghebbenden

In een organisatie zullen verschillende groepen mensen gebruik maken van metadata. Deze groepen hebben allemaal baat bij het gebruik van metadata en zijn hierdoor belanghebbenden. Enkele groepen die in een organisatie gebruik maken van metadata zijn:

- Ontwikkelaars;
- IT project managers
- Eind gebruikers
- DBMS catalog (DBA's)

Bovenstaande groepen gebruiken allemaal metadata bij hun dagelijkse werkzaamheden, maar elke groep kijkt met een andere bril naar de metadata en hebben dus hun eigen eisen. Sommige groepen zullen niet eens weten wat metadata zijn, maar het is wel belangrijk om de verschillende groepen te identificeren. Een eindgebruiker bijvoorbeeld werkt elke dag met "gegevens / informatie" via een soort software interface. Deze informatie krijgt vorm en zijn waarde nadat de data in context is gebracht door middel van metadata. "Jan" zegt niks totdat het in relatie wordt gebracht met "naam" van een klant.

Metadata kunnen in verschillende categorieën onderverdeeld worden [31].

- ***Specifieke metadata***: ·Elke metadata die vereist is door slechts een belanghebbende groep. Deze metadata is zo precies dat het van een bron voorkomt equivalent aan de belanghebbende categorie. Alle metadata die functioneel afzonderlijk is van zowel architectuur als gebruiksperspectief. Voorbeelden van unieke metadata zijn: Tabel primaire sleutel (bron en vereist door DBMS catalog), Case tool versie (bron en vereist door de case tool metadata store), Data Model Creator (bron en vereist door de case tool metadata store);

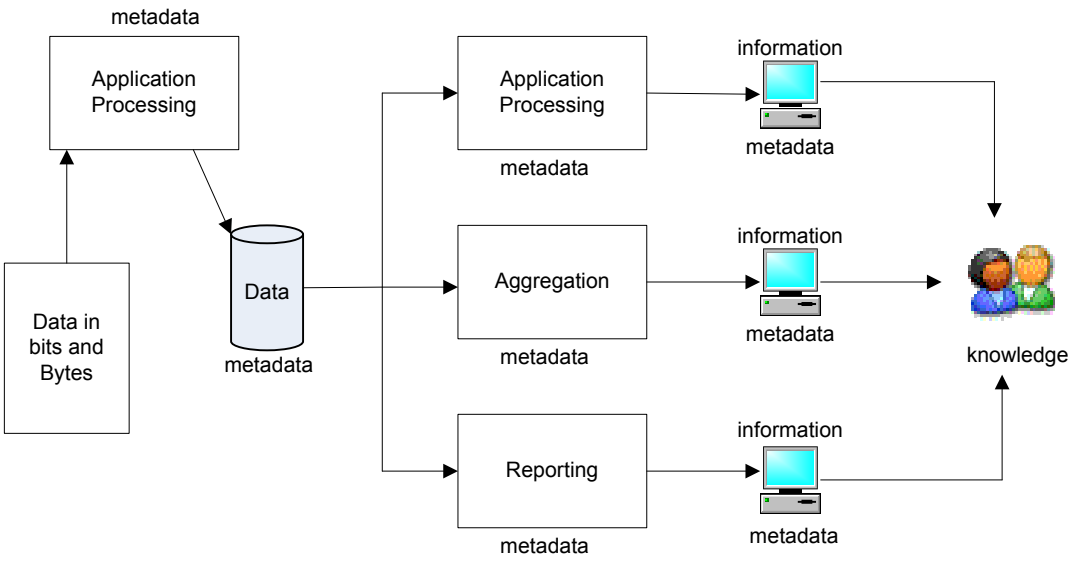
- ***Unieke metadata***: ·Metadata die nodig is maar altijd komt van een totaal andere bron dan de belanghebbende categorie. Deze categorie metadata is nog steeds gewenst, vereist en gebruikt door één belanghebbende groep, maar is altijd van een andere bron. Voorbeeld unieke metadata: Legacy application project manager (is gewenst en vereist door alleen IT project managers, maar komt van oorsprong uit de internal data dictionary;

- ***Algemene metadata***: ·Metadata die door allen is gewenst. Hiermee wordt bedoeld dat de metadata in deze categorie gewenst, vereist en gebruikt wordt door alle belanghebbende groepen. Dit is de metadata die centralisatie en verder uitwerk en standaardisatie vereist.

- ***Niet gespecificeerde metadata***: ·Dit zijn metadata die gewenst zijn door tenminste één belanghebbende groep. Het probleem met dit type metadata is dat elke belanghebbende een eigen terminologie gebruikt. Wat de ene groep een 'data element' noemt, noemt een andere groep een 'veld'. Het is moeilijk om dit soort variaties aan te duiden bij vooral een grote en complexe organisatie, waardoor de metadata niet gespecificeerd wordt. Een ander voorbeeld is wanneer de metadata vanuit verschillende bronnen beschikbaar is en hierdoor niet gespecificeerd wordt.

3.3 Waarom is metadata belangrijk?

De term informatie wordt buiten de IT wereld vaak gedefinieerd als kennis. Naarmate informatie technologie (IT) begon te groeien, werd het normaal om de termen kennis en informatie te scheiden, wat vroeger niet het geval was. De enen en nullen die we verwerken noemen wij nu data, applicaties verwerkt de data in informatie en de business gebruikers converteert de informatie in kennis. Metadata spelen een heel belangrijke rol in de evolutie van data naar informatie en informatie naar kennis. Zie ook Figuur 3-2.



Figuur 3-2 Waarde van metadata in organisatie

Metadata komen overal voor in de IT werelden, zoals Middleware moderne applicatie ontwikkeling, Object Oriëntatie, webapplicaties, Product Lifecycle Management enz. Deze verschillende werelden hebben hun eigen terminologie, maar metadata komt voor in al deze werelden en ze zoeken allemaal naar vergelijkbare oplossingen[1]: hoe kan informatie geïntegreerd worden?

Waarom metadata?

- Metadata helpen gebruikers om informatie te vinden / lokaliseren;
- De juiste informatie (snel) te kunnen vinden;
- Metadata helpt bij het identificeren / het in verband brengen van verschillende inhouden (contents);
- Metadata brengen gebruiker en inhoud / informatie samen;
- Verbeteren van informatie retrieval op Internet;
- Toenemen van waarde van informatie als een bezit (asset).

Een belangrijk doel van beschrijvende metadata is om ontdekking / lokalisatie van relevante informatie te vergemakkelijken. Maar metadata kan ook helpen bij het organiseren van elektronische bronnen, vergemakkelijken van interoperabiliteit en integratie van legacy systemen, digitale identificatie en ondersteuning van archivering en behoud van bronnen [17].

De rol van metadata is vaak de ‘verborgen informatie over informatie’, die de context definieert [47].

3.3.1 Integratie

Integratie (In het Latijns: integer, betekent geheel of volledig/compleet) betekent in het algemeen het combineren van delen, zodat ze samen kunnen werken of een geheel kunnen vormen [39]. Met andere woorden, integratie is een proces waarbij verschillende componenten samensmelten tot een geheel. Bij integratie is er sprake van een hechte koppeling en het suggereert dat de informatiebronnen en de metadata dusdanig met elkaar verweven zijn dat zij als één te benaderen zijn [45]. Informatiesystemen worden steeds complexer door verdergaande integratie van voortbrengingsprocessen en afhankelijkheid van nieuwe geavanceerde technologieën.

De volgende vormen van integratie worden onderscheiden [1]:

- Data integratie: is het probleem om data uit verschillende bronnen te combineren en de gebruiker een verenigde view te geven van de data. Dit probleem kan voorkomen wanneer er sprake is van fusies waarbij verschillend organisaties uit verschillende bronnen (databases) data moeten halen. Data integratie speelt zich vooral af bij data warehousing en bij synchronisatie van updates in verschillende databases. Meer en meer leren ontwikkelaars dit op modelniveau aan te pakken, bijvoorbeeld varianten van E-R diagrammen en view integratie-technieken (= het integreren van deelschema's). Data-integratie kent beperkingen en kan alleen als de gegevenslaag gescheiden is van applicaties en userinterfaces. Een effectieve data integratie kan zorgen voor kosten reductie en een vooruitgang in de productiviteit door het vasttellen van consistentheid, correctheid en betrouwbaarheid van data over de hele organisatie. Data integratie is niet een proces zonder risico’s en wordt vaak verergerd door ongelijke bron systemen, legacy data en het onvermogen om records over verschillende data bronnen samen te voegen. Data integratie is niet zomaar het

matchen en linken van data, maar het is ook het behalen van toegang tot de goede data bronnen op de goede tijd. Data integratie wordt ook wel enterprise information integration genoemd;

- Applicatie integratie. In tegenstelling tot data integratie gaat het bij applicatie integratie niet om de data, maar om de verschillende applicaties in een organisatie samen te brengen, integreren tot een geheel. Bij applicatie integratie worden alle applicaties en databases geïntegreerd tot één bedrijfsbrede toepassing die steunt op één geïntegreerde database. De bekendste voorbeelden zijn ERP systemen of maatwerksystemen. Helaas grootschalig, zéér duur en niet eenvoudig in onderhoud. Applicatie integratie is een zeer moeilijke taak. Voor een organisatie die over verschillende applicaties beschikt voor verschillende doeleinden zoals: klanten gegeven, logistiek, medewerker gegevens etc, zal een applicatie integratie uitermate moeilijk en risicovol zijn. Soms kunnen twee versies van dezelfde applicatie niet met elkaar samenwerken na een grote update, laat staan als het gaat om verschillende applicaties.

- Interface en Object integratie. Communicatie tussen verschillende objecten/componenten verloopt via middleware. Hier is sprake van integratie door interoperabiliteit, dwz niet de systemen of de data worden geïntegreerd, maar de communicatie wordt gestandaardiseerd. Modelleren geschiedt steeds meer met behulp van UML en steeds meer object georiënteerd.

In alle voorgenoemde vormen van integratie speelt metadata een centrale rol: integreren van data is integreren van metadata. Integreren van metadata is meer dan alleen kopiëren van de gegevens. Integreren van gegevens uit verschillende bronnen vereist integreren van abstracte en concrete informatie, van vele bronnen en in verschillende talen. De uitdaging is die van "vertalen", van mapping, ofwel van compositie. De uitdaging is om mapping zoveel mogelijk tot één-op-één te brengen, zodat interpretatieproblemen zo klein mogelijk worden [1]. Meer informatie hierover zie [47]

3.3.2 Interoperabiliteit

Interoperabiliteit betreft de technische interfaces die het mogelijk maken om onafhankelijk ontworpen producten aan elkaar te knopen. Interoperabiliteit is de mogelijkheid om data uit te wisselen tussen verschillende systemen met verschillende hardware, software, data structuur en interfaces met een minimaal verlies van gegevens en functionaliteiten. Door gebruik te maken van voorgedefinieerde metadata schemas, shared transfer protocollen en verwijzingen tussen de verschillende schemas, kan er naadloos gezocht worden over het hele netwerk voor de benodigde gegevens [16]. Bij interoperabiliteit is er sprake van een losse koppeling tussen informatiebronnen en metadata. Een extern systeem is verantwoordelijk voor het ontleden van een informatieverzoek en het vertalen (of mappen) ervan naar het doelsysteem.

De uitdaging is om standaarden te ontwerpen die voldoende flexibel en goed gedefinieerd zijn om ingezet te kunnen worden voor het integreren van een brede verzameling legacy en toekomstige systemen.

3.4 Toepassingen van metadata

Zoals eerder aangegeven komt metadata voor in de verschillende IT werelden en zijn metadata gebaseerde technologieën op veel terreinen te vinden zoals [45]:

- Op het web, zoals de "search" en "retrieve" operaties;
- File managers en bestandssystemen; indien metadata aan een bestand is toegevoegd kunnen bestandssystemen "intelligent" met bestanden omgaan;
- Database management systemen; data dictionaries, zonder welke SQL onmogelijk is.
- "Object Oriented class libraries", een set software routines (class definities) die programmeurs direct kunnen gebruiken voor het schrijven van object georiënteerde programma's.

In de volgende subparagrafen worden enkele toepassingen van metadata in het kort beschreven met voorbeelden.

3.4.1 Metadata en documenten

Metadata zijn gestructureerde informatie om een informatie bron te kunnen beschrijven en lokaliseren, gebruiken of te managen (administratieve metadata). Metadata kunnen ook gegevens zijn die een beschrijving geven van een document, zoals titel, auteurs gegevens etc (beschrijvende metadata). In de bibliotheek discipline wordt vaak metadata gebruikt bij het ordening systeem. Dit systeem wordt gebruikt voor het beschrijven van bronnen die gebruikt kunnen worden voor alle soorten objecten, digitaal of niet digitaal.
Een voorbeeld hiervan is de ordening van gestructureerde metadata zoals titel, hoofdstukken en onderwerpen in een bibliotheek catalog. Deze gestructureerde metadata zijn niet alleen om een bepaald boek in de bibliotheek te kunnen vinden, maar het dient ook als een handleiding of index die aangeeft welke onderwerpen in de ontologie van de bibliotheek bestaan. Er kan met de metadata ook bekeken worden welke andere onderwerpen gerelateerd zijn aan dit oorspronkelijke onderwerp.

3.4.2 Metadata en webdocumenten

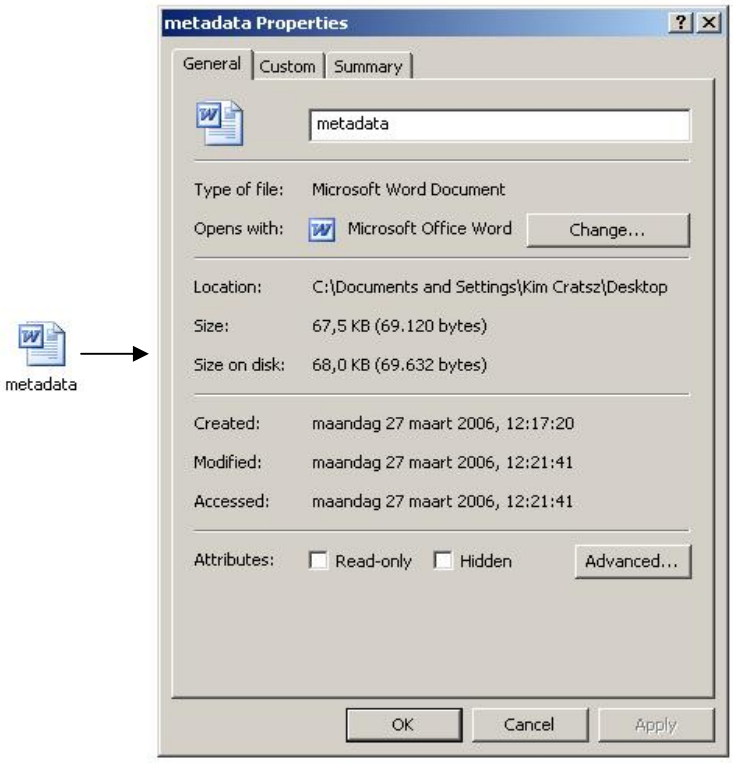
Metadata kunnen binnen een digitaal object zitten zoals webpagina's en met de HTML code meegegeven worden of het kan apart worden opgeslagen. Voorbeeld: metadata wordt vaak in HTML documenten en in de "headers" van foto bestanden gezet.

Tegenwoordig kunnen er allerlei soorten informatie gevonden worden op het Internet. Het probleem wat zich hiermee brengt is dat het moeilijk is om relevante en belangrijke informatie te vinden tussen al deze massa aan informatie. Hierdoor zijn metadata heel erg belangrijk geworden op het World Wide Web. Metadata geven meer waarde, omdat het consistentie creëert. Als een webpagina over een bepaald onderwerp een woord of een zin bevat, dan zouden bij alle andere webpagina's over hetzelfde onderwerp ook dezelfde woord of zin terug moeten komen. Tevens kan het voorkomen dat een onderwerp bekend is onder verschillende namen. Vooral omdat bijvoorbeeld de naam van een sport in verschillende landen net anders wordt genoemd. Metadata maken het mogelijk dat bij het zoeken op een bepaald onderwerp, de relevante webpagina's over dit onderwerp getoond worden. Zelfs als het onderwerp bekend is onder verschillende namen. Meer informatie hierover is te vinden in paragraaf 3.5.

3.4.3 Bestandssysteem metadata

Bijna alle bestandssystemen houden metadata bij van bestanden. Sommige systemen hebben metadata in de directory entries en soms zelfs in de naam van het bestand. De naam van een bestand is eigenlijk al (beschrijvende) metadata, dit geldt ook voor de eigenschappen van het bestand zoals tijdstip van aanmaak en wijziging (administratieve metadata). Het wordt gemakkelijker om bestanden te zoeken aan de hand van de metadata. Enkele voorbeelden van bestandssystemen zijn FAT en NTFS. Het nieuwe bestandssysteem van Microsoft Windows, WinFS, is niet een geheel nieuw bestandssysteem, maar een metadata indeling die draait als dienst bovenop het bestaande bestandssysteem NTFS. In Figuur 3-3 wordt een voorbeeld gegeven van metadata van een bestand [49].

In Figuur 3-3 is er gebruik gemaakt van een simpele word bestand. Dit bestand heeft enkele eigenschappen zoals aangegeven in het grijze menu. Bij de metadata eigenschappen is te zien wat de naam en formaat is van het bestand maar ook de dag en tijdstip waarop het bestand gemaakt en gewijzigd is, dit zijn de administratieve metadata.



Figuur 3-3 Bestandssysteem metadata

3.4.4 Data warehouse metadata

Data warehouse metadata systemen zijn soms verdeeld in twee secties [2]:

- Back room metadata dat gebruikt wordt voor het transformeren, laden van functies om OLTP[1] data te verkrijgen in een data warehouse.
- Front room metadata wordt gebruikt om screens te labelen en het uitdraaien van rapporten.

Voorbeelden van backroom en frontroom metadata zijn in Figuur 3-4 weergegeven.

Category	Example	Category	Example
Back-End	✔ Ownership descriptions of each source schemas ✔ Source file layouts and target schema designs ✔ Definitions and characteristics of tables and columns ✔ Primary/foreign key assignment scheme and relationships ✔ Database partition and disk striping specifications ✔ Index and view definitions and specifications ✔ Mainframe or source system job specifications ✔ COBOL/JCL, C or Basic code to implement extraction ✔ Update frequencies of the original sources ✔ Job specifications for joining sources, stripping out fields, and looking up attributes ✔ Data cleaning, enhancement, and transformation rules, specifications, and mappings ✔ Data audit records ans transformation run-time logs ✔ Access methods, access rights, privileges and passwords for source access	**Front-End**	✔ Process flows ✔ Presentation graphics, e.g., Powerpoint ✔ Flowcharts and program code for accessing source system data ✔ Ownership and Business name of data elements ✔ Business-rule based definitions of data elements ✔ Ownership and Business descriptions of the source systems ✔ Descriptions of the valid values in categorical fields ✔ Descriptions and flowcharts of aggregation and transformation processes ✔ Physical data models ✔ Table join guidance, including cautions and restrictions ✔ Validation statistics for quality control ✔ Legal limitations on usage ✔ User login profiles and security/access controls

Figuur 3-4 Voorbeeld metadata Front- en backroom

Bij de front- en backroom komen gestructureerde, administratieve en beschrijvende metadata voor. Bij de backroom is er sprake van meer administratief en beschrijvende metadata. Bij de frontroom komt meer gestructureerd en beschrijvende metadata voor. Dit betekent niet dat er bij de frontroom geen administratieve metadata voor kan komen.

[1] Online Transaction Processing is a form of transaction processing conducted via computer network. Some applications of OLTP include electronic banking, order processing, employee time clock systems, e-commerce, and eTrading

Zo zijn er bij de front- en backroom, administratieve metadata aanwezig die te maken hebben met rechten van gebruikers (access methods bij backroom en access controls bij frontroom).
Een verschil is dat er bij de back-end met meer administratieve metadata gewerkt wordt dan bij de front-end. Dit is ook logisch, omdat een gebruiker bij de front-end niets te maken heeft met het geven van rechten om bijvoorbeeld in te kunnen loggen. Een gebruiker van de front-end aan de andere kant heeft wel gegevens nodig zoals proces flows etc. Dit zijn gegevens die door gebruikers op de front-end gebruikt worden om hun werkzaamheden te kunnen verrichten, zoals managers.

3.4.5 Relationele database metadata

Elk relationele database systeem heeft een eigen mechanisme voor het opslaan van metadata. Voorbeelden hiervan zijn [21]:

- Tabellen van alle tabellen in een database, naam, grootte en aantal records / rijen in elke tabel.
- Tabellen van kolommen in elke database. In welke tabellen de kolommen worden gebruikt en het type data dat opgeslagen kan worden in elke kolom.

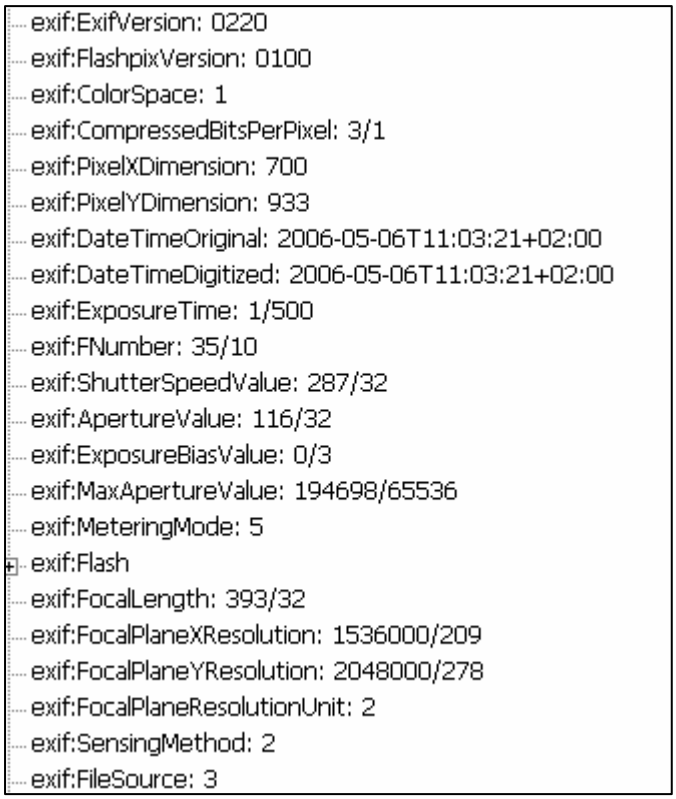
In database terminologie wordt dit soort metadata een “catalog” genoemd. De SQL standaard (vanaf SQL-92) specificeert een uniforme manier om deze catalog te benaderen, genaamd de “information_schema”. Maar niet alle databases implementeren dit information schema. Meer informatie over dit onderwerp is te vinden in hoofdstuk 4, systeem tabellen.

3.4.6 Image metadata

Voorbeelden van foto bestanden die metadata bevatten, zijn EXIF (Exchangeable Image File Format) en TIFF (Tagged Image File Format). EXIF is een specificatie voor het formaat van het image bestand dat gebruikt wordt door digitale camera’s [37]. Deze specificatie gebruikt bestaande bestand formaten zoals JPEG, TIFF REV. 6.0 met een toevoeging van specifieke metadata tags. De soorten administratieve metadata die deze fotobestanden bevatten zijn bijvoorbeeld:

- Datum en tijd informatie (administratief); digitale camera’s houden de datum en tijd bij het maken van een foto bij en slaan deze op als metadata.
- Instelling van de camera; statische informatie als camera model en informatie die varieert bij elke foto zoals de oriëntatie, ISO instelling, diafragma en sluitertijd. Deze gegevens worden ook als administratieve metadata opgeslagen.
- Locatie informatie; vanaf 2004 ondersteunen sommige camera’s de functie om locatie informatie op te slaan door middel van een GPS ontvanger.
- Copyright informatie wordt ook opgeslagen als metadata.

In Figuur 3-5 wordt een voorbeeld gegeven van de EXIF metadata van een foto in Adobe Photoshop CS. Bij image metadata is er ook sprake van beschrijvende metadata, zoals de naam van het bestand.



Figuur 3-5 Voorbeeld EXIF metadata

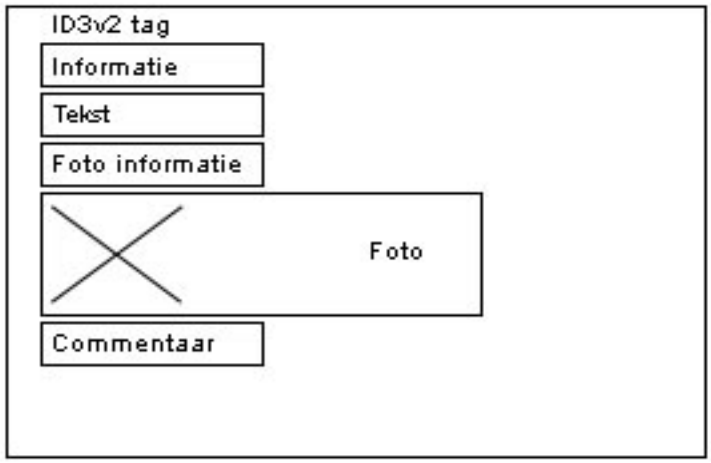
Alle eigenschappen van een foto worden opgeslagen door de camera. Zo is de resolutie van de foto te zien bij de eigenschappen. exif:PixelXDimension: 2048 geeft aan dat de breedte van de foto 2048 pixel is en exif:PixelYDimension: 1536 geeft de lengte van de foto weer. Ook de versie van EXIf wordt aangegeven, in dit geval is dit EXIF versie 2.2. EXIF behoort tot de DCF (Design rule for Camera File system) standaard, ontworpen door JEITA (Japan Electronics and Information Technology Industries Association). DCF is een ISO specificatie die de bestandssystemen voor digitale camera's definieert. Het doel is om interoperabiliteit te motiveren tussen verschillende foto apparaten.

3.4.7 MP3 bestanden

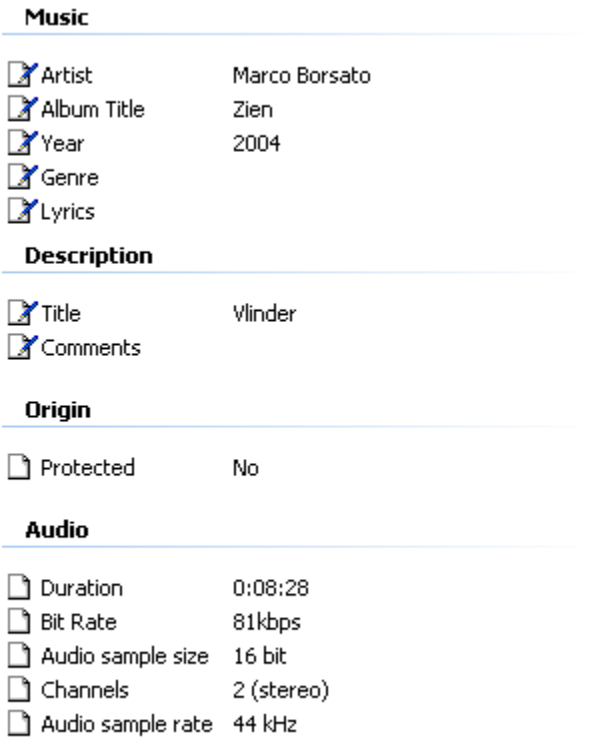
Mp3 bestanden hebben ook metadata tags in een formaat genoemd ID3. ID3 staat toe dat metadata zoals titel, artiest, album en tracknummer opgeslagen wordt in het MP3 bestand. Bij het afspelen van een mp3 muziekbestand wordt vaak de naam van het lied, artiest, album enz weergegeven in het afspeelprogramma. Deze gegevens worden ID3 tags genoemd. Tegenwoordig bestaan er allerlei programma's voor het aanpassen van ID3 tags, dit kan handig zijn wanneer de ID3 tag informatie niet correct is [41].

Andere muziekbestand formaten gebruiken of een eigen meta tag of helemaal geen tag. ID3 tag is speciaal ontworpen voor mp3 bestanden. Zo heeft WMA een eigen tag formaat voor het opslaan van metadata.

Een voorbeeld van hoe de lay-out van de ID3 tag eruit ziet is weergegeven in Figuur 3-6.



Figuur 3-6 Layout ID3v2 tag



Figuur 3-7 Voorbeeld eigenschappen van een MP3 file

.

In Figuur 3-7 zijn de eigenschappen van een MP3 file te zien. De eigenschappen bij het kopje music en description bestaan uit beschrijvende metadata. Bij origin en audio is er sprake van administratieve metadata.

De laatste ID3 versie is de ID3v2 tag. Bij elke tag wordt er stukken informatie opgeslagen, genoemd frames. Een voorbeeld van enkele voorgedefinieerde frames wordt hieronder weergegeven.

Tabel 3-1 ID3 frames

BPM (beats per minute)	File type	Year of the recording
Composer	Time of recording	Part of a set
Content type	Content group description	Original release year
Copyright message	Title/songname/content	Internet radio station owner
Date of recording	Initial key	File owner/licensee
Playlist delay	Original filename	Publisher
Encoded by	Length	Size
Lyricist/Text writer	Media type of audio original	Recording dates

3.5 Metadata schemes

Metadata schemes of schema zijn sets van metadata elementen, ontworpen voor een specifiek doel, bijvoorbeeld om een specifiek type informatie object te beschrijven. De definitie of betekenis van een element zelf is de semantiek van de scheme en de waarden die aan de metadata elementen gegeven worden zijn de inhoud. Metadata schemes specificeren meestal de namen van de elementen en hun semantiek, maar soms worden de inhoud regels voor hoe een inhoud geformuleerd moet worden ook gespecificeerd, bijvoorbeeld hoe de titel geïdentificeerd moet worden. Er kunnen ook syntax regels zijn voor hoe de elementen en hun inhoud gecodeerd moeten worden. Een metadata scheme zonder voorgeschreven syntax regels wordt syntax onafhankelijk genoemd [16].

3.6 Metadata standaarden.

Tegenwoordig bestaan er verschillende metadata schemes standaarden. In de volgende paragrafen volgen enkele van deze standaarden.

3.6.1 Metadata Encoding and Transmission Standard (METS)

De METS schema is een standaard voor codering van beschrijvende, administratieve en gestructureerde metadata betreffende objecten in een digitale bibliotheek door gebruik te maken van de XML schema taal van het World Wide Web consortium [43].

METS was ontworpen om de noodzaak voor een standaard data structuur voor beschrijving van complexe digitale bibliotheek objecten mogelijk te maken [17].

Een METS document bestaat uit zeven secties:

- METS Header: ·Informatie die de METS document zelf beschrijft zoals de maker en redacteur;
- Descriptive metadata: ·Dit is de metadata die het object beschrijft (beschrijvende metadata);
- Administrative metadata ·Dit is de administratieve metadata zoals de url.
- File section · Informatie die bijhoudt welke versies van het object aanwezig zijn. Bijvoorbeeld in mp3 en WAV formaat.
- Structural map · Geeft de hiërarchische structuur van het object weer. Bijvoorbeeld een intro, een tussenstuk en een einde.
- Structural links ·Legt vast welke hyperlinks er allemaal aanwezig is in een structuur map. Bijvoorbeeld een foto die eigenlijk een links is naar een ander document.
- Behavior: ·Kan gebruikt worden om executable behavior te associëren met inhoud in een METS object

3.6.2 Metadata Object Description Schema (MODS)

Een ander voorbeeld van een metadata schema is Metadata Object Description Schema. MODS is een beschrijvende metadata schema en is afgeleidt van MARC21. MARC21 formaat is een standaard voor de presentatie en communicatie van bibliografische records en gerelateerde informatie in machine leesbare vorm. Deze metadata schema maakt gebruik van XML en is voornamelijk bedoeld voor bibliotheek applicaties, maar kan ook voor andere doeleinden gebruikt worden. Een voorbeeld van een dergelijk schema wordt in Figuur 3-8 gegeven.

MODS kan gebruikt worden als een extensie schema van METS, maar ook voor originele bron omschrijving in XML syntax of presentatie van een MARC21 record in XML.

In Figuur 3-8 wordt het MODS meta schema gebruikt. Bij deze schema wordt er gebruik gemaakt van XML om een object te beschrijven, in dit geval deze scriptie. Zo is te zien dat de titel van de scriptie wordt aangegeven (bij title), de naam van de auteur, uitgeverij jaar van uitgave, plaats en ISBN nummer.

Een MODS Record voorbeeld

```
<mods>
    <titleInfo>
        <title>Demystifying Metadata & Metamodel</title>
    </titleInfo>
    <name type="personal">
        <namePart type="family">Cratsz</namePart>
        <namePart type="given">Kim</namePart>
        <role>
            <roleTerm authority="marcelator" type="text">author</roleTerm>
        </role>
    </name>
    <typeOfResource>text</typeOfResource>
    <originInfo>
        <dateIssued>2006</dateIssued>
        <place>
            <placeTerm type="text">Arnhem, GLD</placeTerm>
        </place>
        <publisher>KUN</publisher>
    </originInfo>
    <identifier type="isbn">1-234567-89-0</identifier>
</mods>
```

Figuur 3-8 MODS voorbeeld

3.6.3 Dublin Core

De Dublin Core metadata element set is ontstaan na een discussie bij een workshop in 1995. De workshop was gehouden in Dublin, Ohio en hierdoor is de elementen set, de Dublin Core genoemd. De Dublin Core Metadata Initiative, (DCMI), bestuurt de voortdurende ontwikkeling van de Dublin Core en gerelateerde specificaties [36].

De originele doelstelling van Dublin Core was om een set van elementen te definiëren, die gebruikt konden worden door auteurs om hun eigen webbronnen te beschrijven.

Door de snelle groei van elektronische bronnen werd het catalogiseren van al deze bronnen bijna niet meer te doen door bibliotheken. Het doel van Dublin core werd toen om een paar elementen en een paar simpele regels te definiëren die toegepast konden worden door leken. Dit doel resulteerde in dertien Core elementen. Deze originele (13) Core elementen zijn later tot vijftien elementen uitgebreid. Deze 15 elementen worden de Dublin Core Metadata Element Set (DCMES) genoemd. DCMES is een set van 15 beschrijvende semantiek definities en was gecreëerd om een core set van elementen aan te bieden die door verschillende disciplines gebruikt kan worden of door organisaties die de noodzaak hebben om informatie te organiseren of te classificeren.

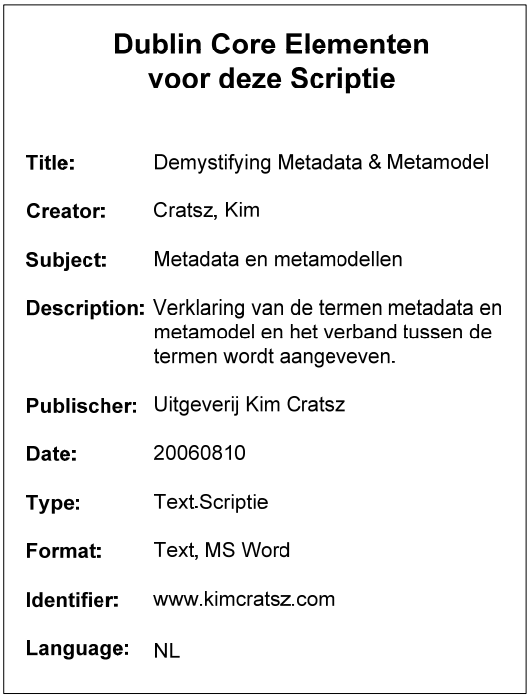
De Dublin Core was ontworpen om simpel en beknopt te zijn, en om web gebaseerde documenten te beschrijven. Maar de Dublin Core werd ook gebruikt bij grotere en

complexere applicaties. Hierdoor is de Dublin Core gesplitst in gekwalificeerde en simpele (niet gekwalificeerde) Dublin Core. De kwalificering wordt gebruikt om een element te verfijnen of om de codering scheme te identificeren in representatie van een element waarde [16]. De simpele Dublin Core gebruikt geen kwalificering en bestaat uit de 15 elementen. Deze 15 elementen zijn: titel, maker, onderwerp, beschrijving, uitgeverij, contributor, datum, type, formaat, identificeren, bron, taal, relatie, coverage en rechten. Een uitgebreide omschrijving van de 15 elementen is te vinden in tabel 2.2 “de 15 Core elementen“.

Doordat de Dublin Core eenvoudig is bij gebruik, is deze standaard gebruikt bij verschillende andere toepassingen, zoals onderzoekers, muziek verzamelaars etc. Voor meer informatie over de Dublin Core, zie [36].

Iedereen kan gebruik maken van de Dublin Core metadata voor het beschrijven van de bronnen van een informatie systeem. Web pagina’s zijn een van de meest gebruikte type bron, die gebruik maken van de Dublin Core elementen om een object te beschrijven. Dublin Core metadata wordt gebruikt als de basis voor beschrijvende systemen door verschillende gemeenschappen zoals:

- Opleidingsinstituten;
- Bibliotheken;
- Regeringsinstituten;
- Wetenschappelijk onderzoek sector;
- Organisaties die gebruik maken van websites.



Figuur 3-9 Dublin Core voorbeeld

.

In Figuur 3-9 is er van deze scriptie een Dublin Core voorbeeld gegeven. Er worden tien elementen van de Dublin Core gebruikt, waaronder titel, maker, onderwerp, omschrijving, uitgever, datum, type, formaat, identifier en taal.

Tabel 3-2 De 15 core elementen van Dublin Core

Element Name: Title	
Label:	Title
Definition:	A name given to the resource.
Comment:	Typically, Title will be a name by which the resource is formally known.
Element Name: Creator	
Label:	Creator
Definition:	An entity primarily responsible for making the content of the resource.
Comment:	Examples of Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
Element Name: Subject	
Label:	Subject and Keywords
Definition:	A topic of the content of the resource.
Comment:	Typically, Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource.
Element Name: Description	
Label:	Description
Definition:	An account of the content of the resource.
Comment:	Examples of Description include, but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.
Element Name: Publisher	
Label:	Publisher
Definition:	An entity responsible for making the resource available
Comment:	Examples of Publisher include a person, an organization, or a service. Typically, the name of a Publisher should be used to indicate the entity.
Element Name: Contributor	
Label:	Contributor
Definition:	An entity responsible for making contributions to the content of the resource.
Comment:	Examples of Contributor include a person, an organization, or a service. Typically, the name of a Contributor should be used to indicate the entity.
Element Name: Date	
Label:	Date
Definition:	A date of an event in the lifecycle of the resource.
Comment:	Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and includes (among others) dates of the form YYYY-MM-DD.
Element Name: Type	
Label:	Resource Type
Definition:	The nature or genre of the content of the resource.
Comment:	Type includes terms describing general categories, functions, genres, or aggregation levels for content. To describe the physical or digital manifestation of the resource, use the FORMAT element.

Element Name: Format	
Label:	Format
Definition:	The physical or digital manifestation of the resource.
Comment:	Typically, Format may include the media-type or dimensions of the resource. Format may be used to identify the software, hardware, or other equipment needed to display or operate the resource. Examples of dimensions include size and duration.
Element Name: Identifier	
Label:	Resource Identifier
Definition:	An unambiguous reference to the resource within a given context.
Comment:	Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Formal identification systems include but are not limited to the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).
Element Name: Source	
Label:	Source
Definition:	A Reference to a resource from which the present resource is derived.
Comment:	The present resource may be derived from the Source resource in whole or in part. Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system.
Element Name: Language	
Label:	Language
Definition:	A language of the intellectual content of the resource.
Comment:	Recommended best practice is to use RFC 3066 which, in conjunction with ISO639 [ISO639]), defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English, "akk" for Akkadian", and "en-GB" for English used in the United Kingdom.
Element Name: Relation	
Label:	Relation
Definition:	A reference to a related resource.
Comment:	Recommended best practice is to identify the referenced resource by means of a string or number conforming to a formal identification system.
Element Name: Coverage	
Label:	Coverage
Definition:	The extent or scope of the content of the resource.
Comment:	Typically, Coverage will include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and to use, where appropriate, named places or time periods in preference to numeric identifiers such as sets of coordinates or date ranges.
Element Name: Rights	
Label:	Rights Management
Definition:	Information about rights held in and over the resource.
Comment:	Typically, Rights will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions may be made about any rights held in or over the resource.

4. Systeem tabellen

In het voorgaande hoofdstuk is uitgelegd dat metadata de eigenschappen zijn van een object zoals een document. Om de term metadata beter uit te leggen zijn er verschillende praktische voorbeelden gegeven waar metadata toegepast wordt. Enkele voorbeelden hiervan waren Mp3 bestanden, documenten, foto bestanden etc. Ook relationele databases is genoemd als voorbeeld bij toepassingen van metadata. Maar waar is deze metadata precies te vinden? In de tabellen of kolommen van de database? Hoofdstuk vier zal hierop een antwoord geven.

Hoe gaan de verschillende soorten database systemen om met metadata? De data dat de structuur van de database beschrijft? Er zijn verschillende soorten database systemen en ze hebben allemaal een andere manier of eigenlijk een andere naam voor de term metadata. Door middel van de DBMS (Database Management System) kan een gebruiker zien wat voor tabellen, kolommen en restricties (constraints) de database heeft. Het verschil is dat SQL server dit de system tables noemt, IBM noemt het “the system catalog” en oracle de “data dictionary”. Deze systemen ondersteunen allemaal de functie om systeem tabellen te kunnen benaderen en bekijken, alleen de manier/instructie om de metadata gegevens te kunnen raadplegen is anders.

In standaard SQL wordt de data die de database beschrijft (metadata) het information_schema genoemd. Met andere woorden een DBMS die de SQL standaard gebruikt kan door middel van de SELECT statement informatie over de structuur van de database uit het information_schema halen.

4.1 Catalogus

Elke database heeft een set van systeem catalogus tabellen die de logische en fysieke structuur van de data omschrijven. Deze tabellen bevatten verschillende informatie zoals; definities van de database objecten, de beschrijvende metadata zoals de gebruikers tabellen, views, indexen, relatie namen, attribuut namen, attribuut domeinen (data typen), constraints (sleutel, verwijzende sleutel) en de informatie over beveiliging zoals de autorisatie die de gebruikers hebben over deze objecten (administratieve metadata) [21]. Deze tabellen worden gecreëerd bij het maken van de database en worden bijgewerkt gedurende de normale operaties. Deze tabellen kunnen niet expliciet worden gemaakt of verwijderd, maar een gebruiker kan ze wel bekijken en raadplegen door gebruik te maken van een query en een catalog view.

Een SQL product houdt niet alleen gegevens over tabellen en kolommen bij, maar ook een lijsten met namen en wachtwoorden van gebruikers en de volgorde waarin kolommen met de CREATE TABLE instructie gecreëerd zijn, dit zijn de administratieve metadata.[8]. Met andere woorden, SQL beschikt over tabellen waarin metadata wordt opgeslagen voor eigen gebruik. Deze tabellen met metadata worden catalogustabellen genoemd en vormen samen de catalogus. Het systeem catalogus is zelf een minidatabase en één van zijn voornaamste functies is om de schemas of beschrijving van de database die de DBMS heeft op te slaan. Het bevat een beschrijving van de conceptuele database schema (basis tabellen), de internal schema (beschrijving van de storage en de index), de external schema (view definities) en de mappings tussen de schemas op de verschillende niveaus.

Elke catalogustabel is een gewone tabel die met SELECT instructies geraadpleegd kan worden, maar kunnen niet met UPDATE of DELETE benaderd worden. Het is ook niet gewenst dat de catalogus direct gewijzigd of verwijderd kan worden. Als dit het geval was, dan zou een gebruiker een hele database kunnen verwijderen met alle belangrijke gegevens. Daarom is het ook belangrijk dat niet iedereen toegang kan krijgen tot deze tabellen. Zie ook paragraaf 4.4 “beveiligen catalogustabellen”.

Hieronder volgen enkele voorbeelden van de catalogustabellen en wat deze bevatten [21]:

Tabellen	Bevat voor elke tabel onder andere het tijdstip waarop de tabel gemaakt is en de eigenaar (gebruiker die de tabel gecreëerd heeft).
Kolommen	Bevat voor elke kolom het datatype, de tabel waartoe het behoort, of de NULL waarde wel of niet is toegestaan en het volgnummer van de kolom in de tabel.
Indexes	Bevat voor elke index de tabel en de kolommen waarop de index gedefinieerd is en de wijze waarop de index gesorteerd is.
Views	Bevat voor elke view de view definitie (de SELECT instructie)
Gebruikers	Bevat per gebruiker de naam en het wachtwoord.

4.2 Query

Een gebruiker kan informatie uit de catalogus halen door middel van een query. Hieronder wordt een voorbeeld gegeven van een query waarbij er gegevens worden gehaald uit de kolommen met tabel naam “students” en die gecreëerd is door de SQL database administrator. De kolom naam, data type en kolom nummer worden met deze SELECT instructie gerepresenteerd in de view.

```
SELECT      COLUMN_NAME, DATA_TYPE, COLUMN_NR
FROM        COLUMNS
WHERE       TABLE_NAME = 'STUDENTS'
AND         TABLE _CREATOR = 'SQLDBA'
ORDER BY    COLUMN_NR;
```

4.3 Raadplegen van de catalogustabellen

Elke DBMS heeft een catalogus waarin de gegevens van de tabellen worden geregistreerd. Zonder de catalogus wordt het voor het systeem niet mogelijk om te bepalen of een vraag door een gebruiker wel of niet beantwoord mag en kan worden. Elke database systeem is anders en heeft ook een andere benaming voor de catalogus.

Echter niet alle database systemen stellen de catalogus ter beschikking voor de gebruikers. Systemen met SQL als databasetaal bieden wel de mogelijkheid en de instructies om de catalogus te kunnen raadplegen en bekijken. Catalogustabellen worden in SQL als normale tabellen weergegeven [9]

4.3.1 Functie raadplegen catalogustabellen

Het raadplegen van de catalogustabellen kan verschillende functies hebben. Hieronder volgen er een paar functies:

Helpfunctie	Voor een nieuwe gebruiker kan het bijvoorbeeld handig zijn om te bepalen welke tabellen en kolommen in de database aanwezig zijn.
Controlefunctie	Indien een bepaalde tabel verwijderd gaat worden, kan een gebruiker door het raadplegen van de catalogustabellen bepalen welke indexen, views en bevoegdheden ook verwijderd zullen worden met de tabel.

4.4 Beveiligen van de catalogustabellen:

De catalogustabellen dienen net zoals andere normale tabellen beveiligd te worden. Bij normale tabellen kan door middel van de GRANT instructie aangegeven worden welke gebruikers allemaal toegang hebben tot een tabel, hiermee wordt een netwerk van bevoegdheden opgezet. De GRANT instructie kan op dezelfde manier ook gebruikt worden voor de catalogustabellen. De catalogustabellen bevatten belangrijke gegevens die zeker niet voor iedereen toegankelijk behoren te zijn. Doorsnee gebruikers behoren bijvoorbeeld geen toegang te hebben tot USERS tabel waarin alle namen met wachtwoorden geregistreerd staan. Als dit wel het geval zou zijn, zou de gehele beveiliging geen waarde meer hebben.

4.5 Waarom catalog?

Waarom is een information schema belangrijk in een database systeem? Waarom hebben de verschillende database systemen een catalogus? Het incorporeren van een catalogus in een database systeem brengt structuur, maar waarom is deze structuur nodig voor een database systeem en wie heeft dit bedacht?

4.5.1 The twelve rules

In 1970 heeft Dr. Edgar F. Codd een theorie gestart over relationele databases. Dr. Codd wordt ook de vader van de relationele databases genoemd, omdat hij de eerste man was die met deze theorie kwam. Voor 1970 waren er nog geen database systemen gebaseerd op het relationele model en vijftien jaar later werd zijn artikel genaamd "The Twelve Rules" gepubliceerd. In dit artikel geeft Dr. Codd een beknopt verhaal over de ideale relationele database met dertien regels [34]. De vierde regel is:

"*Active online catalog based on the relational model.*
The system is required to support an online, inline, relational catalog that is accessible to authorized users by means of their regular query language". That is, users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data [35].
Deze vierde regel geeft aan dat een relationele database toegang moet kunnen geven tot zijn structuur door gebruik te maken van dezelfde instrumenten die gebruikt worden om de "normale" data te benaderen. Dit kan worden bereikt door de structuur op te slaan in de zogenaamde speciale systeem tabellen ofwel de catalogus. Met inline of in-line wordt bedoeld dat de catalog direct beschikbaar is in de database. Het is geen link naar een andere database of document waar de catalog zich bevindt, maar het is actief, up to date en beschikbaar in de database zelf.

Met deze vierde regel van Dr. Codd wordt het noodzakelijk om met een DBMS de omschrijving van de database op dezelfde manier zoals gewone data die opgeslagen is in de database te representeren. Dit maakt het mogelijk voor gebruikers om informatie uit de catalogus te bekijken door gebruik te maken van dezelfde databasetaal die ze gebruiken voor het bekijken van normale data in de database. Het voordeel hiervan is dat gebruikers alleen maar één databasetaal moeten leren om alle benodigde gegevens te kunnen benaderen en bekijken.

Het systeem catalog is een perfect voorbeeld van een "in-line" metadata. Een wijziging van de in-line metadata heeft een directe impact op de definitie van het ontwerp, structuur, toegankelijkheid van de data en de processen die de data gebruiken. Een wijziging van een tabel index in het systeem catalog zal de actuele index van de tabel wijzigen. Hierdoor kan het wijzigen van de data in het systeem catalog heel erg riskant zijn en mogen alleen administrators deze rechten hebben.

4.5.2 Voordelen catalogus

Op het Internet zijn er nog paar voordelen van de vierde regel genoemd [24]:

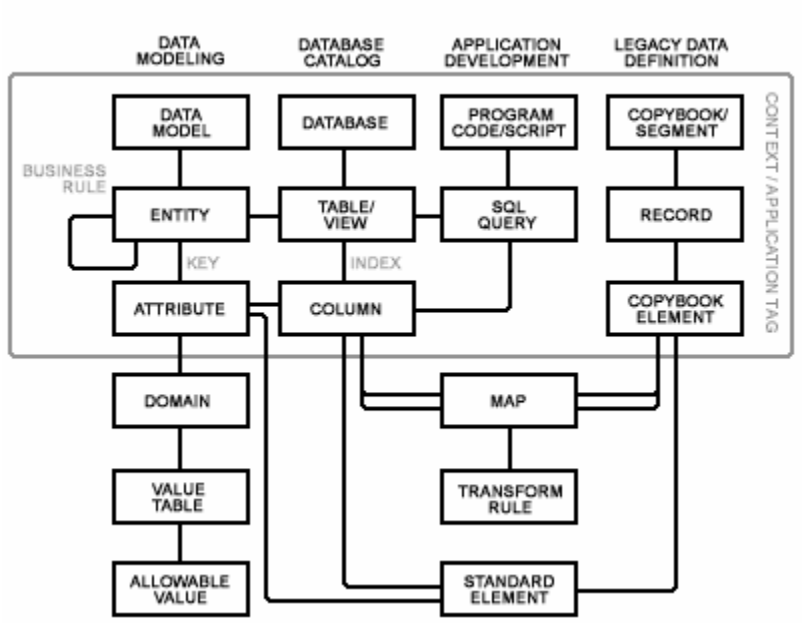
- Een gebruiker hoeft geen nieuwe termen of keywords te leren om metadata informatie op te halen, de gebruiker kent de SELECT instructie al.

- Er zijn verschillende manieren om de resultaten van een query te selecteren en te presenteren in vergelijking met de LIST TABLES instructie van de oude DB2 commando, "sp_help" van een oude SQL Serverprocedure of SQLTables() de functie van de ODBC lagoon.

- Bijna elke DBMS maakt gebruik van de catalogus en de mogelijke SQL instructies om de gegevens te bekijken. Hierdoor ontstaat er portabiliteit.

Waarom is metadata voor de database administrator belangrijk en waarom zou de administrator geïnteresseerd zijn in metadata? In de praktijk begrijpen de administrators het belang van de data die beschikbaar is in het systeem catalog, de metadata. Een administrator zal altijd gebruik maken van de metadata net zoals een werknemer in de organisatie data / gegevens nodig heeft om zijn dagelijkse werkzaamheden te kunnen verrichten. Beide personen hebben data / gegevens nodig voor het beoefenen van hun dagelijkse werk. Voor de administrator is de metadata de data die belangrijk is voor zijn werk.

De metadata in het systeem catalogus staat centraal in Figuur 4-1. Core Metamodel. De database, tabellen, views van data, indexen en de kolommen (gerepresenteerd als metadata typen in het diagram) zijn de belangrijkste schakel in de definitie van metadata die de organisatie ondersteunt.



Figuur 4-1Core Metamodel [26]

De data die opgeslagen wordt in een database heeft geen waarde als het niet in relatie wordt gezet met elkaar. Een organisatie gebruikt de beschikbare gegevens / data die ze hebben om hun processen te kunnen ondersteunen. Zonder DBA (database administrator) metadata is er geen verbinding tussen de data die de enterprise ondersteunt en de gewone data. Zonder DBA metadata is er geen relatie / verbinding tussen de SQL en de data. Zonder DBA metadata is er geen enterprise structuur van metadata op technisch niveau [27]. In Figuur 4-1 is te zien dat de data modeling, Application development en legacy definition allemaal verbonden zijn door middel van de database catalog, de metadata. Dit geeft aan hoe belangrijk de metadata is in een organisatie. De metadata staan centraal.

Het is bijna te simpel om te denken dat metadata van het systeem catalog een grote impact kan hebben op het vermogen om de enterprise data te managen, maar dit is wel

het geval. Het gaat niet om de catalog zelf, maar om de data die in de database en catalog aanwezig is, het vormt de basis voor de organisatie.

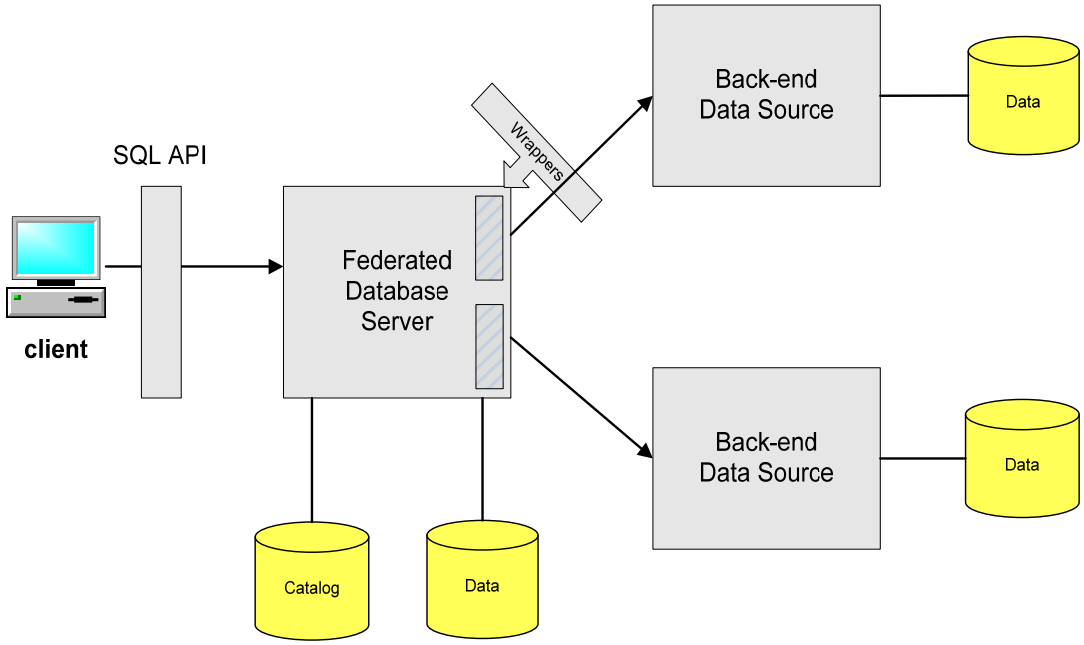
4.6 Federale databases

In grote organisaties is het soms bijna niet te vermijden dat verschillende gedeelten van de organisatie verschillende databases zullen gebruiken voor het opslaan van data. Soms zijn er ook organisaties die verschillende filialen hebben die over de hele wereld gevestigd zijn. Deze filialen beschikken dan over hun eigen database. Dit betekent dat in al deze verschillende databases enorm veel data wordt opgeslagen. Het is pas als al de data van deze verschillende systemen bij elkaar wordt gebracht dat de organisatie (enterprise) de gehele waarde van de data kan inzien en gebruiken. In dergelijke situaties kan een organisatie ervoor kiezen om een federale database server te gebruiken [40].

Een federale database systeem (FDBS) is een soort metadatabase management systeem dat uit verschillende autonome (sub) database systemen bestaat. Deze verschillende sub databases worden in één federale database geïntegreerd. De verschillende databases zijn in verbinding via een computer netwerk en kan geografisch gedecentraliseerd zijn. Aangezien de verschillende databases anoniem zijn, is een federale database in tegenstelling tot het samenvoegen van de verschilleden databases (wat soms een afschrikweekend proces kan zijn) een goed alternatief [38].

Federale database systemen kunnen door een abstractie van de data, een uniforme interface bieden aan de eindgebruiker. De eindgebruiker wordt in staat gesteld om data op te halen en op te slaan vanuit verschillende databases door middel van één query en zelfs als er sprake is van heterogene databases. Om dit mogelijk te maken, zal de federale database systeem de query moeten deconstrueren in verschillende sub queries die door de verschillende DBMS's verwerkt kan worden. Doordat verschillende database management systemen verschillende query talen gebruiken, kan een federale database systeem van een wrapper gebruik maken om de sub queries te vertalen in de geschikte query taal. De wrapper definieert de communicatie mechanisme die gebruikt zal worden om toegang te krijgen tot de data via een remote data source.

In Figuur 4-2 is de architectuur van de IBM federale systeem weergegeven.



Figuur 4-2 Architectuur van een IBM federale systeem [40]

Problemen die bij het implementeren van een FDBS kunnen voorkomen zijn onverenigbare data types en query syntax. Maar dit is niet het enige obstakel dat voor kan komen. Een generiek probleem ligt in het matchen van semantiek equivalent, maar verschillende benamingen van verschillende schemas (data modellen ➔ tabellen en attributen). Federale database systemen is ook een vorm van integratie, zie ook paragraaf 3.3.1 en 4.6.2.

4.6.1 Voorbeeld probleem: gebruik van verschillende databases

In Figuur 4-3 wordt van drie verschillende winkels de verkoop cijfers in tabelvorm gedemonstreerd [50]. Alle drie de winkels slaan de verkoop gegevens op in een eigen database en op een andere manier. De winkel in Indiana maakt gebruik van één tabel met drie kolommen (figuur 4.3.A). De winkel in Chicago maakt ook gebruik van één tabel, maar deze tabel heeft 5 kolommen (figuur 4.3.B). De derde winkel is gelegen in Milwaukee en deze maakt gebruik van drie tabellen met elk drie kolommen om dezelfde gegevens als de andere twee winkels op te slaan (figuur 4.3.C). Om de verkoop gegevens van alle drie de winkels met elkaar te kunnen vergelijken is er een query nodig die per database de gegevens op een andere manier moet gaan benaderen. Met een query zal het uiteindelijk wel lukken om de verkoopcijfers van de verschillende systemen met elkaar te vergelijken, omdat het hier om een vrij simpele database gaat. Wel kan een wijziging in een tabel van één van de databases als gevolg hebben dat de query niet meer het gewenste resultaat zal opleveren of misschien zelfs niet meer werkt.

Figuur 4.3.A: Database Indiana

Sales

Store	Dept	AvgSales
PineSt	Wmn	62500
WestRd	Wmn	75000
AndAve	Wmn	81500
PineSt	Wmn	50000
AndAve	Wmn	73500
PineSt	Toddler	41250
WestRd	Toddler	55000
AndAve	Toddler	68500

Figuur 4.3.B: Database Chicago

AvgSales

Store	Wmn	Men	Boy	Girl
CedarRd	48500	35000	25500	
CtrSq	55500	50000	32000	52500
WashSt	63500	58500	42250	58500
IllStt	78000	63250	50000	65500

Figuur 4.3.C: Database Milwaukee

LincAveSales

Dept	AvgSales	LCode
Girl	45000	1
Boy	55000	2
Men	65000	3
Wmn	80000	4

FreePkSales

Dept	AvgSales	WCode
Boy	28500	B38
Men	46500	C18
Wmn	60000	X27

WashStSales

Dept	AvgSales	FCode
Girl	35000	A
Men	48500	B
Wmn	55000	C

Figuur 4-3 Gemiddelde verkoop cijfers uit 3 databases

De gegevens in Figuur 4-3 kunnen door middel van een query nog bekeken en vergeleken worden omdat de tabellen nog redelijk eenvoudig zijn en het niet veel data betreft. Bij grotere databases waarbij er sprake is van veel meer tabellen en veel meer data en data typen is het veel moeilijker en misschien zelfs onmogelijk om de gegevens uit verschillende databases met elkaar te gaan vergelijken. In dit geval zal een query het werk niet meer kunnen doen, maar zullen er verschillende queries met sub queries nodig zijn. Vooral grote organisaties met verschillende filialen die gebruik maken van een eigen database hebben te maken met dit soort problemen, maar ook als er sprake is van een fusie kunnen organisaties hiermee te maken krijgen.

4.6.2 Schema afhankelijk of onafhankelijk

Het voorbeeld gegeven in Figuur 4-3 geeft op een simpele manier aan hoe organisaties met verschillende systemen / databases moeten werken. Een federale database systeem kan een oplossing zijn voor dit probleem. Maar het opzetten van een federale database systeem is niet een eenvoudig proces. Zoals eerder uitgelegd is het matchen van verschillende schemas een groot obstakel bij het opzetten van een FDBS. Hoe kan via een query een database component van een MDBS worden aangeroepen, die semantiek gelijke data opslaat maar gebruik maakt van een ongelijk schema [19]?

Een gebruikelijke manier voor het matchen van verschillende schema's is om van elke database een lokaal schema te maken. Van deze lokale schema's wordt vervolgens een globaal schema gemaakt waarop de FDBS is gebaseerd. Deze techniek maakt gebruik van het schema afhankelijke benadering en wordt ook wel de "schema integration approach" genoemd. Deze techniek heeft echter een aantal problemen:

1) Het matchen van lokale schemas tot een globaal schema is een heel erg ingewikkeld proces. Bij een grote organisatie waarbij er sprake is van honderden databases zal het matchen van de schema's misschien zelfs onmogelijk worden. Bij aanvullende (complementary) databases, (dit zijn databases met ongelijke soort data die bij samenvoeging elkaar aanvullen), kan deze techniek voor het matchen van schemas wel geschikt zijn.

2). Deze techniek voor het matchen van schema's brengt een ander probleem met zich mee. Indien een lokaal schema gewijzigd wordt, dan zullen de verschillende queries niet meer werken. Aangezien de metadata in een database dynamisch is, dit betekent het verandert over tijd, zullen de queries met de tijd ook opnieuw herschreven moeten worden.

Een andere oplossing voor het matchen van metadata bij een FDBS, is de schema onafhankelijke benadering. Een schema onafhankelijke benadering verzekert dat alle queries geldig blijven onder elke metadata wijziging (wijziging in het lokale schema). Een techniek waarbij er gebruik wordt gemaakt van deze benadering is het "schema coordination approach". Het schema coordination approach is geschikt voor complementary en competing databases (competing databases zijn databases met gelijke soort data). Voor meer informatie over het schema coordination en integration approach zie [51].

Een voorbeeld van schema afhankelijk is weergegeven in Figuur 4-4. Figuur 4-4A geeft de database schema van de inkoop weer. Figuur 4-4B is de database schema van het magazijn. Bij beide schema's komt "material" voor. Bij inkoop komt "material" als een attribuut voor en bij magazijn als een entiteit. Dit is een structureel conflict. Verder zijn er ook naam conflicten te herkennen. Bij de inkoop database is het entiteit "unit", semantiek equivalent aan de "parts" entiteit bij het magazijn schema. Dit geldt ook voor de attributen "serial#" en "part_ID".

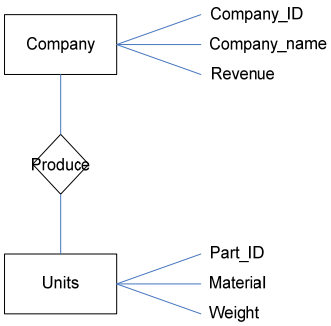


Fig. 4-4A: De inkoop database schema

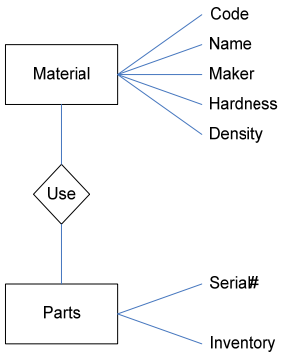


Fig. 4-4B: Het magazijn database schema

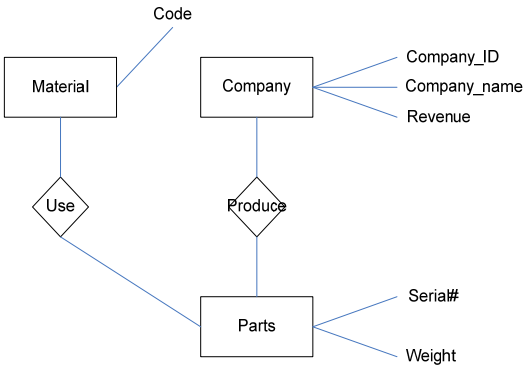


Fig 4-4C: De gewijzigde inkoop database schema

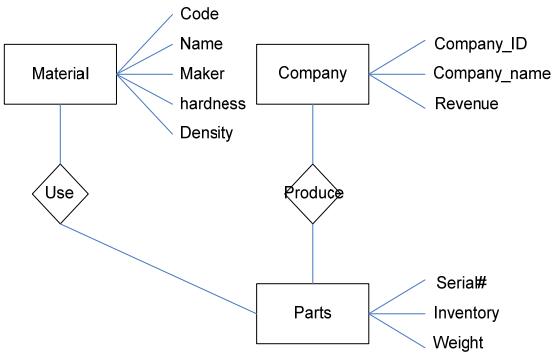


Fig. 4-4D: De geintegreerde database schema

Figuur 4-4 Voorbeeld schema afhankelijk [51]

Figuur 4-4C geeft het begin van de integratie tussen de twee schema's weer. De naam van de "Units" entiteit wordt gewijzigd in "Parts". Tevens wordt ook de naam van "Pasrt_ID" gewijzigd in "Serial#". Vervolgens kan het attribuut "Material" meegenomen worden bij entiteit "Material" met als attribuut "code". Figuur 4-4D laat het eindresultaat van de integratie van de twee schema's zien. Dit voorbeeld is nog redelijk makkelijk, omdat het niet verschillende grote database betreft met veel meer entiteiten en attributen. Hoe ingewikkelder de databases, hoe moeilijk het integratie proces tussen de verschillende schemas wordt.

4.7 Eigenschappen in tabellen

Zoals eerder uitgelegd in hoofdstuk 3, kan metadata omschreven worden als de eigenschappen van een gegeven informatie zowel fysiek als elektronisch. Een gegeven informatie is in dit geval een object en de metadata zijn de eigenschappen van het object.

De structuur kan in de vorm van (O, D) worden aangegeven waarbij O een object is zoals een document en D de data is over het object in kwestie zoals titel en auteur.

In Figuur 4-5 wordt een voorbeeld gegeven van een database met een tabel student. Deze tabel heeft in totaal vijf kolommen, namelijk STUDENTNR, STUDENTNAAM, ADRES, WOONPLAATS en TELEFOONNR. Deze kolommen zijn eigenschappen van het object Student. Gebaseerd op de definitie van metadata zijn deze gegevens (de kolommen) eigenlijk metadata van het object student.

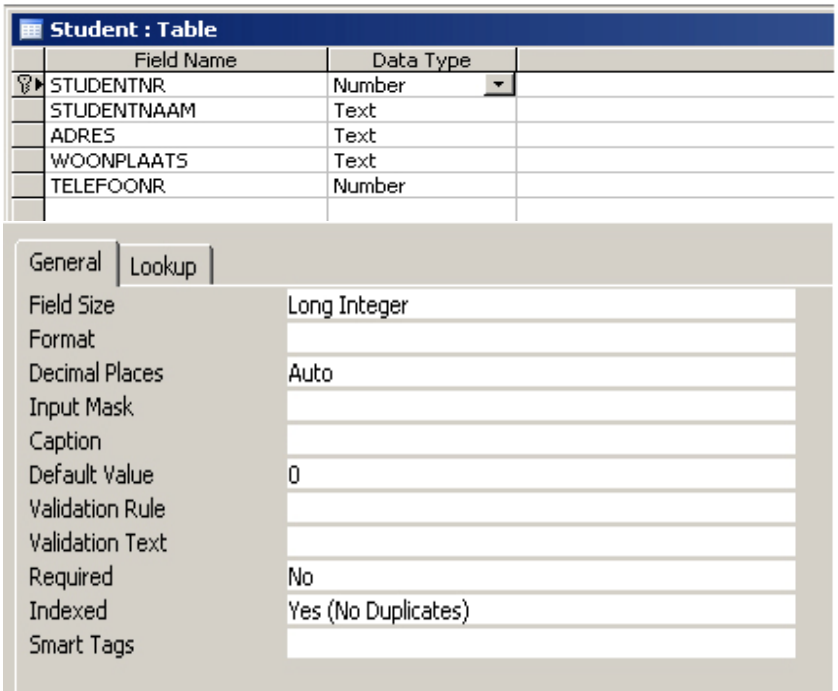
TABEL: STUDENT

STUDENTNR	STUDENTNAAM	ADRES	WOONPLAATS	TELEFOONNR
1	Jan	Straat 1	Zutphen	1234
2	Peter	Straat 2	Arnhem	5678
3	Klaas	Straat 3	Nijmegen	9101
4	Willem	Straat 4	Ede	1213

Figuur 4-5 Tabel Student

De kolommen kennen attributen, zoals kolom STUDENTNR = integer en STUDENTNAAM = string enz. Deze gegevens zijn ook metadata, omdat ze de tabel beschrijven. Dit betekent dat de metadata in een database niet alleen via de catalogus benaderd kan worden, maar ook gewoon in de database aanwezig is.

De attributen van een kolom zijn te zien in Figuur 4-6. De gegevens zoals data type = “Number” en Field Size = “Long Integer” zijn allemaal metadata omdat ze de tabel beschrijven.



Figuur 4-6 Attributen en kolommen als metadata

5. Metamodel

Net zoals bij metadata, zijn er ook veel vragen over wat metamodellen zijn en wordt de term ook op verschillende manieren opgeschreven zoals; “meta-model”, “meta model” en “metamodel”. In deze scriptie is ervoor gekozen om “metamodel” te hanteren. Maar wat is een metamodel precies en wat kan men ermee doen, wat is het nut ervan?

De meest simpele definitie van een metamodel is een model van bijvoorbeeld een modelleringstaal of eigenlijk een model van een model. De term meta geeft aan dat het over “iets” gaat, maar meta betekent dat het hoger is. Met andere woorden een metamodel benadrukt het feit dat een metamodel de modelleringstaal beschrijft op een hoger abstractie niveau dan de modelleringstaal zelf. Het abstraheren is het weghalen van het overbodige.

Een metamodel is ook een model, maar niet hetzelfde model als de modelleringstaal zelf. Het metamodel beschrijft de essentiële eigenschappen van de taal, die gemodelleerd wordt, zoals de concrete (zie 4.1) en abstracte syntax (zie 4.2), en semantiek (zie 4.3) van de taal. Een metamodel is een precieze definitie van de constructies en regels die nodig zijn voor het creëren van semantiek modellen [6].

5.1 Concrete syntax

Alle talen bieden een notatie die de presentatie en constructie van modellen of programma’s in de talen vergemakkelijken. Deze notatie wordt de concrete syntax genoemd en kan onderverdeeld worden in tekstuele syntax en visuele syntax.

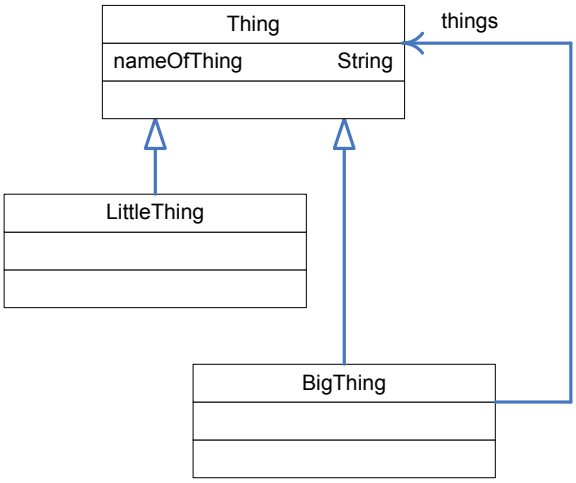
5.1.1 Tekstuele syntax

Tekstuele syntax maakt het mogelijk om modellen of programma’s te beschrijven in een gestructureerde en tekstuele vorm. Hieronder volgt een voorbeeld van een tekstuele syntax [6].

```
public abstract class Thing
    {
    private String nameOfThing;
    public String getName()
            {return nameOfThing;}
    }
```

5.1.2 Visuele syntax

Visuele syntax presenteert een model of programma in een diagram vorm. Een visuele syntax bestaat uit een aantal grafische iconen die een view representeert van het onderliggende model. In Figuur 5-1 wordt een voorbeeld gegeven van een klasse in een visuele syntax [6].



Figuur 5-1 Voorbeeld van een klasse in visuele syntax vorm

5.2 Abstracte syntax

De abstracte syntax beschrijft de woordenschat van concepten van een taal en hoe ze gebruikt kunnen worden om modellen te creëren. Het bestaat uit definities van de concepten, de relaties die tussen de concepten bestaan en regels die aangeven hoe de concepten officieel mogen worden gecombineerd. Een abstracte syntax van een state machine taal kan bijvoorbeeld de volgende concepten bevatten; state, transition en event [6].

5.3 Semantiek

De abstracte syntax geeft aan welke concepten in een taal kunnen voorkomen, maar de abstracte syntax beschrijft niet wat deze concepten precies betekenen. Wat een concept precies inhoudt is belangrijk voor de modelleur. Als een concept niet goed wordt begrepen kan het ook op verschillende manieren worden toegepast die niet correct zijn. Om ambiguïteit te voorkomen wordt er gebruik gemaakt van de semantiek [6].

Vragen zoals; wat is een state, wat betekent een state, hoe kan een transitie gebeuren, zijn allemaal vragen die door de semantiek van de taal kan worden beantwoord. Hierdoor is het noodzakelijk dat de semantiek van een taal duidelijk en compleet is [44].

5.4 Modelleringstechnieken

Modelleringtechnieken worden gebruikt voor de specificatie van verschillende aspecten van een informatie systeem. Één methode kan bestaan uit verschillende technieken. Een methode zoals Information Engineering past verschillende technieken toe, waaronder ER diagrammen (Entity Relationship Diagrams) voor data modelleren en structure charts voor de structuur van programma's.

Elke techniek heeft een eigen taal voor het modelleren van bijvoorbeeld het domein of UOD (Universe of Discourse) zoals eerder beschreven in dit hoofdstuk. Elke techniek, of eigenlijk elke modelleringstaal is gebaseerd op een eigen metamodel. Het metamodel van een techniek is echter een model dat de aspecten beschrijft die onafhankelijk zijn van het domein/UoD die beschreven zijn door de techniek in een specifiek geval.

De beschrijving van een model, dat afhankelijk is voor een bepaalde situatie/case wordt een applicatiemodel genoemd. Een applicatiemodel is een instantie van het metamodel van de techniek die gebruikt wordt voor het beschrijven van het applicatiemodel. Een metamodel beschrijft dus een model en mogelijk de relaties tussen andere modellen [22].

De definitie van metamodel: ***"metamodel is een model gemaakt op een hoger abstractie niveau die de regels en constructies (syntax en semantiek) van het onderliggende model aangeeft".***

5.5 Waarom metamodellen

Metamodelleren is een activiteit die onder andere metamodellen produceert. Bij het modelleren zoals bij ORM of Object georiënteerde modellering worden modellen geproduceerd. Bij metamodellering worden ook modellen geproduceerd, maar deze zijn metamodellen. Metamodellen en modellen zijn een poging om de wereld om ons heen te beschrijven voor een bepaald doel.

Metamodellen worden gemaakt voor een bepaald doel net zoals modellen ook gemaakt worden voor een bepaald doel. Zo zijn er modellen die gespecialiseerd zijn in het modelleren van processen en anderen in het modelleren van informatie. Het is hierdoor belangrijk om te weten wat het doel is van het metamodel. Als het doel veranderd, dan kan dit grote gevolgen hebben voor het metamodel. Als het doel van ORM wordt gewijzigd om ook de processen van een bedrijf in kaart te kunnen brengen, dan zal het metamodel ook aangepast moeten worden.

Een metamodel beschrijft de essentiële eigenschappen van de taal die gemodelleerd wordt. Deze eigenschappen zijn de abstracte syntax en de semantiek van de taal. Een metamodel is dus een precieze definitie van de constructies en de regels. Als het doel verandert, dan veranderen ook de regels en constructies en zal het metamodel aangepast moeten worden.

Metamodellen kunnen gebruikt worden voor verschillende doeleinden [48]:

- Metamodel als een schema voor de semantiek gegevens, dat uitgewisseld moet worden. (Uitwisseling van CASE tools gegevens).
- Als een taal die een specifieke methodologie of proces ondersteunt. Dit was het originele doel van het UML metamodel. Later is UML ook gebruikt voor data uitwisseling en opslag. Dit heeft tot veel problemen in het metamodel geleidt waardoor het aangepast moest worden.
- Als een taal om additionele semantiek van een bestaande informatie uit te drukken. Voorbeeld is om informatie op het WEB meer berekenbaar te maken.
- Metamodel om te "mappen" naar andere modelleringstalen, zoals van ORM naar UML of omgekeerd.

Het begrip semantiek is heel erg belangrijk voor metamodellen. Het reflecteert de noodzaak om niet alleen "iets" te modelleren in de echte wereld, maar om de betekenis of doel dat dit "iets" heeft voor het doel van het metamodel.

Als een persoon een vierkant tekent in een teken applicatie, dan moet de applicatie kunnen begrijpen dat het een vierkant is. Maar als deze applicatie gebruikt wordt om de workflow van een bedrijf in kaart te brengen, dan is een vierkant geen simpele vierkant meer maar dan bijvoorbeeld een proces zoals van een inkoop afdeling. Twee personen kunnen naar het vierkantje kijken en een heel andere interpretatie hebben van de betekenis van het vierkant. Dit betekent dat deze twee personen ook een heel andere metamodel in gedachte hebben [25].

Een metamodel is dus een collectie van concepten (dingen termen etc). Deze concepten zijn de woordenschat waarmee je praat over een bepaald domein, zoals cirkel, diagram, proces, transitie etc. Het is heel erg belangrijk om precies te zijn wanneer de betekenis van "iets" wordt beschreven.

5.6 Onderverdeling modellen[5]

Een model is een representatie van een systeem. Een model kan antwoorden geven op vragen in plaats van het systeem. Mogelijke vragen die gesteld kunnen worden, hangt af van het systeem. De set van mogelijke vragen die gesteld kunnen worden, is een subset van de set van vragen die gesteld kunnen worden aan het systeem. Zie hiervoor

Figuur 5-2. Deze beperking is het gevolg van het feit dat een model geen uitgebreide beschrijving is van het systeem, maar alleen een representatie geeft van sommige aspecten die gemodelleerd zijn. De set van aspecten is gespecificeerd door het metamodel. Het metamodel definieert een set van concepten en relaties.

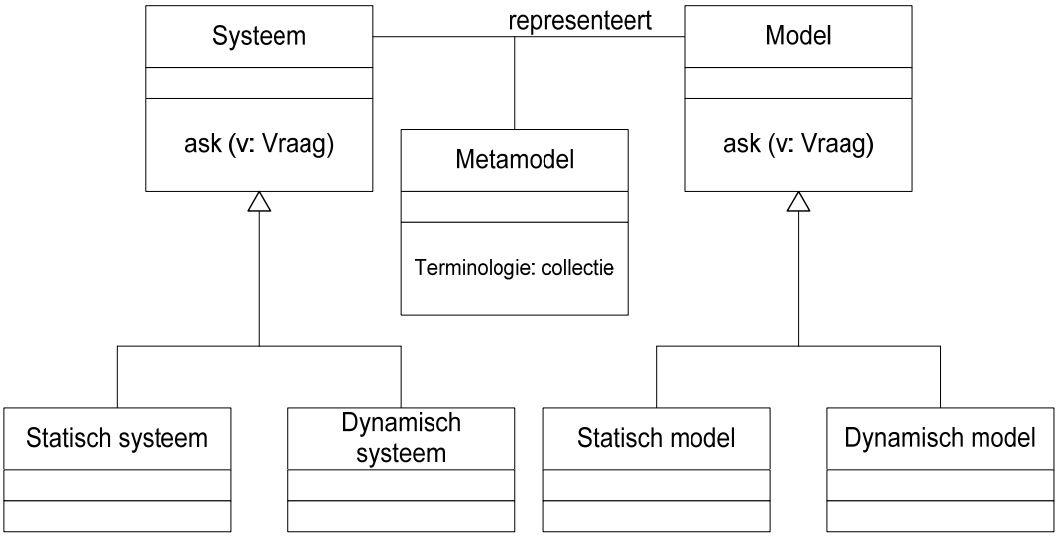
5.6.1 Statisch en dynamisch [5]

Een systeem kan gerepresenteerd worden door verschillende modellen en elk model is gebaseerd op een verschillend metamodel. Er kan een verschil worden gemaakt tussen statische en dynamische systemen. Een statisch systeem is een systeem die zich niet kan ontwikkelen. Een statisch systeem is dus een "snapshot", het kan niet wijzigen met de tijd. Een vraag stellen aan een statisch systeem op verschillende momenten zal hetzelfde antwoord opleveren. Dynamische systemen kunnen in vergelijking met statische

systemen wel wijzigen. Een vraag zoals: “hoeveel mensen wonen in Nederland?” kan op verschillende momenten verschillende antwoorden geven. De reden hiervoor is dat er elke dag mensen geboren worden en dood kunnen gaan. Het antwoord op de vraag zal hierdoor op verschillende momenten (waarschijnlijk) anders zijn. Het antwoord op de vraag reflecteert de huidige situatie op het moment dat de vraag wordt gesteld. Doordat er statisch en dynamische systemen zijn, zijn er ook statische en dynamische modellen.

Een statisch model is een model dat niet kan wijzigen, het is een vaste representatie van het systeem. Wanneer een systeem statisch is, dan is het model ook statisch. Van een dynamisch systeem kan ook een statisch model gemaakt worden, maar dit model zal dan alleen bepaalde gevallen die altijd waar zijn van het systeem representeren. Een dynamisch model kan de wijzigingen die in een systeem plaats vinden reflecteren.

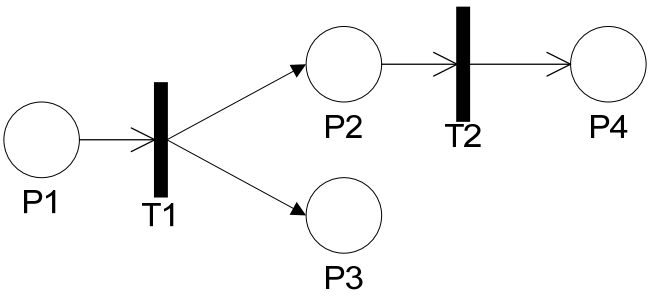
Statische modellen kunnen alleen vragen van het type “ wat is” beantwoorden. Dynamische modellen kunnen ook dit type vragen beantwoorden, maar het is interessanter om te weten wat het antwoord is op vragen wanneer een bepaalde stimulans is gegeven. Dit betekent dat dynamische systemen niet alleen vragen van het type “wat is” kunnen beantwoorden, maar ook van het type “wat als”, de “if question”.



Figuur 5-2 Onderverdeling van systeem en model

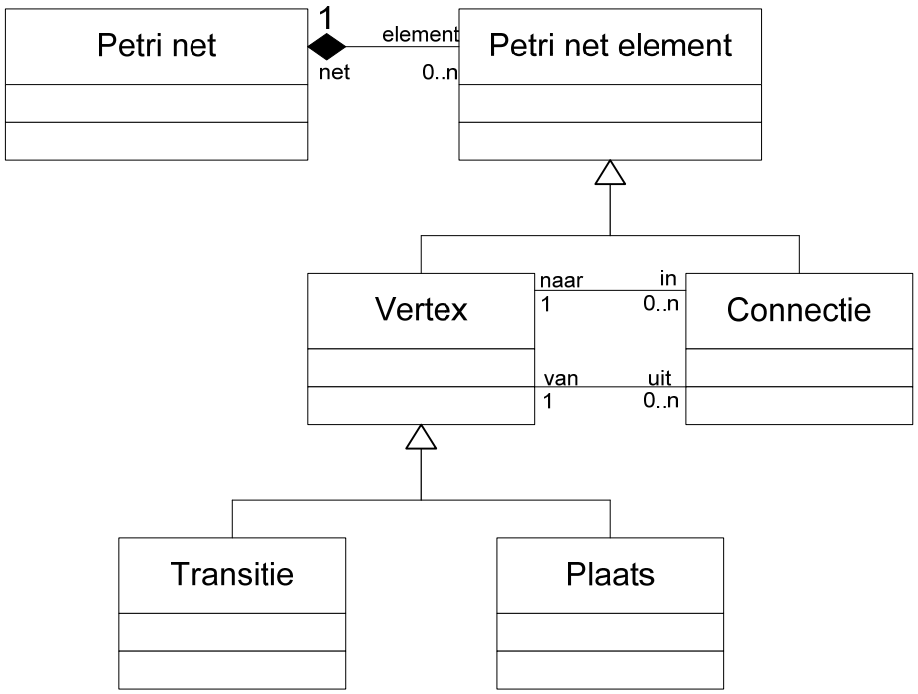
De set van concepten die gemanipuleerd kunnen worden in een model, zijn gedefinieerd in het metamodel. Hierdoor hangt het af van het metamodel of een model statisch of dynamisch is. Een metamodel voor relationele databases, die beperkt is tot tabellen en kolommen, zal alleen een statische representatie toestaan.

Een voorbeeld van een dynamisch model zal worden weergegeven in Petri nets. Zie hiervoor Figuur 5-3.



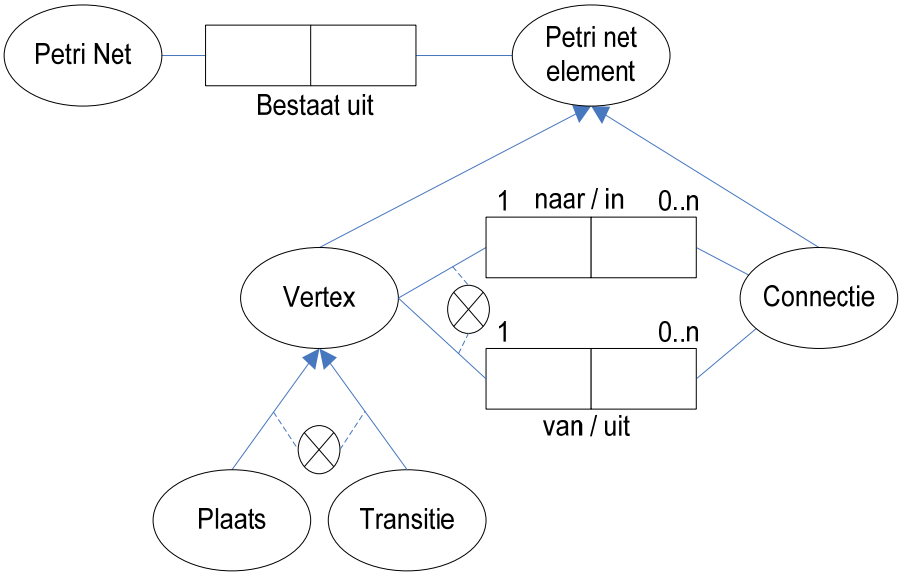
Figuur 5-3 Voorbeeld dynamisch model in Petri nets

In Figuur 5-3 is een voorbeeld gegeven van een dynamisch model. Dit dynamische model is gemaakt in Petri nets. Een simpel petri net is een set van plaatsen (beschreven door cirkels Pn) welke met elkaar verbonden zijn via transities (Tn) door middel van connecties / pijlen. Een connectie / pijl gaat van een plaats naar een transitie of van een transitie naar een plaats. Bovenstaand figuur heeft vier plaatsen (P1, P2, P3 en P4) en twee transities (T1 en T2).



Figuur 5-4 Een mogelijk metamodel van Petri net

Een Petri net bestaat uit Petri net elementen. Deze elementen kunnen of connecties of vertices zijn. Een vertex is of een transitie of een plaats. Het voorbeeld in Figuur 5-3 is een model die gebaseerd kan zijn op het metamodel gegeven in Figuur 5-4.



Figuur 5-5 Metamodel van Petri net in ORM

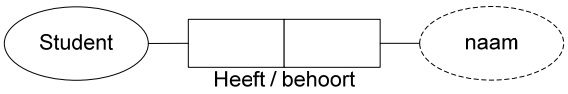
In Figuur 5-5 is een begin gemaakt van een metamodel van Petri net in ORM. Het is dus wel mogelijk om van een modelleringstaal of applicatiemodel een metamodel te maken in een andere taal dan de modelleringstaal zelf. Van het mogelijke metamodel van Petri net zoals weergegeven in Figuur 5-4, is een mogelijk metamodel gemaakt zoals weergegeven in Figuur 5-5, maar van ORM.

5.6.2 Soorten modellen en metamodellen

Modellen kunnen door iedereen gemaakt worden en ze kunnen over alles gaan. Zoals eerder uitgelegd, is een model een interpretatie van de modelleur. Twee verschillende personen kunnen van hetzelfde domein twee verschillende modellen maken, omdat ze vanuit verschillende invalshoeken naar het domein kijken. Dit betekent dat er ook verschillende soorten metamodellen gemaakt kunnen worden, gebaseerd op de perceptie en representatie van een modelleur [25]. De modellen zijn dan allemaal een instantie van het gemaakte metamodel.

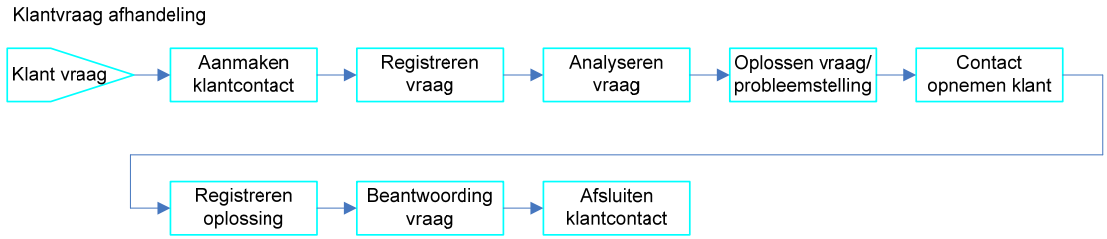
Eerder is uitgelegd dat er een verschil is tussen een statisch en een dynamisch model. Verder kan er geconcludeerd worden dat er een metamodel gemaakt kan worden van een modelleringstaal en van een applicatiemodel. Bij beide modellen kan er een onderscheidt gemaakt worden tussen het datamodel en het procesmodel. Een datamodel is een model die op een abstracte manier beschrijft hoe de data is gerepresenteerd in een organisatie. Het doel van procesmodellen is om processen te documenteren en communiceren van bijvoorbeeld een organisatie. Figuur 5-6 en Figuur 5-7 geeft van beide soorten modellen een voorbeeld.

Datamodel:



Figuur 5-6 Simpel voorbeeld van een datamodel in ORM

Procesmodel:

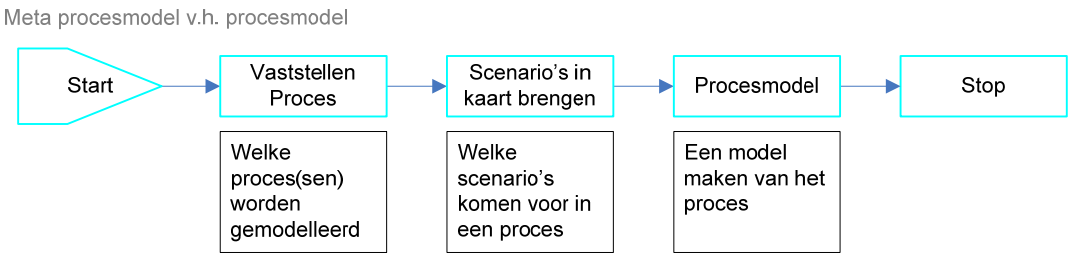


Figuur 5-7 Simpel voorbeeld van een procesmodel van een vraag afhandeling

Figuur 5-6 is een simpel voorbeeld van een datamodel gemaakt in ORM. Het procesmodel zoals weergegeven in Figuur 5-7 is een voorbeeld van een mogelijk proces binnen een organisatie voor het afhandelen van een klantvraag. Zaken zoals hoe een klant zijn vraag kan stellen (persoonlijk, telefonisch, Internet) en of er altijd een oplossing of antwoord gegeven kan worden of dat de vraag door een andere afdeling beantwoord moet worden, zijn buiten beschouwing gelaten. Het procesmodel is dus niet een volledig beeld van een werkelijk proces binnen een organisatie, maar een simpel voorbeeld om uit te leggen wat een procesmodel is.

Door de gemaakte onderverdeling van een model in data- en procesmodel, kunnen er vier soorten metamodellen worden onderscheiden:

- Metadatamodel van het datamodel- Dit metamodel beschrijft welke objecten en relaties er in het datamodel voorkomen. Voor een voorbeeld van een metadatamodel van een datamodel zie Figuur 5-9.
- Metadatamodel van het procesmodel- Dit metamodel beschrijft welke objecten en relaties er in het procesmodel voorkomen.
- Metaprocesmodel van het datamodel- Dit metamodel beschrijft in welke stappen het datamodel gemaakt kan worden.
- Metaprocesmodel van het procesmodel- Dit metamodel beschrijft in welke stappen het procesmodel gemaakt kan worden. Een simpel voorbeeld hiervan is gegeven in Figuur 5-8.



Figuur 5-8 Voorbeeld Metaprocesmodel

Figuur 5-8 is een voorbeeld van een Metaprocesmodel. Bij een procesmodel wordt bijvoorbeeld een proces binnen een organisatie in kaart gebracht. Zo is er bij Figuur 5-7 een voorbeeld gegeven van het proces "klantvraag afhandelen", waarbij de behorende activiteiten in het proces gemodelleerd zijn. Een meta procesmodel beschrijft de stappen voor het maken van een procesmodel zoals het proces "klantvraag afhandelen". Van elke proces modelleringstaal kan er een verschillend metamodel gemaakt worden. Bij het voorbeeld van Figuur 5-8 is er gekozen voor een simpel metaprocesmodel om het idee uit te leggen, maar een metaprocesmodel kan er veel ingewikkelder uitzien. In de praktijk bestaan er verschillende talen voor het modelleren van procesmodellen. Enkele van deze talen zijn: Petri nets, EPOS en E3.

Resultaten van het gebruik van metaprocesmodellen zijn een toename van productiviteit van proces modelleurs en een vooruitgang in de kwaliteit van modellen die de metamodellen produceren.

Verder kan er een metamodel gemaakt worden over "alles". Hiermee wordt bedoeld dat een willekeurige persoon niet per se een metamodel moet maken van een applicatiemodel of modelleringstaal. Zoals eerder aangegeven kan er ook een metamodel gemaakt worden van een tabel. Dit betekent dat er een model en dus ook een metamodel gemaakt kan worden door een persoon over alles en van alles.

De taal van het metamodel hoeft ook niet van een specifieke modelleertaal te zijn. Zo kan een persoon met een aantal vierkanten al een model en een metamodel maken, zonder dat het een (de facto standaard) modelleertaal hoeft te zijn.

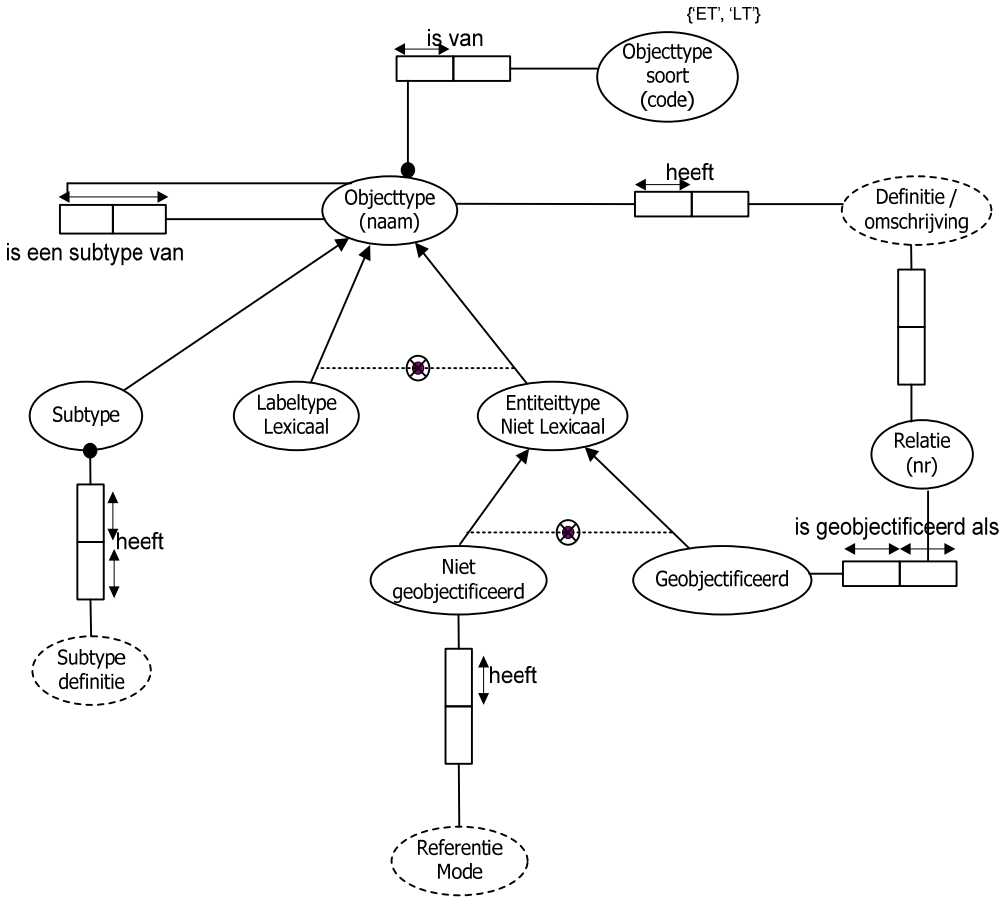
5.7 Voorbeeld metamodellen

Een metamodel van een modelleertaal kan er verschillend uitzien. Zoals eerder uitgelegd heeft dit te maken met de interpretatie van de modelleur die het metamodel gaat maken. Een modelleringstaal zoals ORM heeft bijvoorbeeld verschillende metamodellen omdat er ook verschillende versies bestaan van het modelleertaal ORM. Sommige versies ondersteunen bijvoorbeeld meer beperkingregels (constraints) dan andere versies, waardoor het metamodel er anders uit zal zien. Het verschil in metamodellen kan ook aan de voorkeur van de modelleur zelf liggen. Als een modelleur ervoor kiest of de voorkeur heeft om niet met ternaire feittypen (drie rollen) te werken, dan zal dit ook in het metamodel terug komen. De keuzes van de modelleur heeft grote gevolgen voor het metamodel van een modelleringstaal.

De taal waarmee een modelleur een metamodel wil maken is niet gebonden aan het model. Met andere woorden een metamodel van de modelleringstaal ORM hoeft niet met

ORM gemodelleerd te worden. Het metamodel kan ook in UML of een andere modelleringstaal gemodelleerd zijn. De keuze voor een andere taal bij het modelleren van het metamodel kan het proces moeilijker maken. Vooral als een modelleur een totaal verschillende taal voor het metamodel kiest dat de modelleringstaal zelf. In Figuur 5-3 is een voorbeeld gegeven van een Petri net model. Van dit model werd er in Figuur 5-5 een mogelijk metamodel gemaakt. Dit metamodel is niet gemaakt met de taal Petri net, maar met de taal UML, een totaal andere taal dan dat er gebruikt is voor het maken van het model.

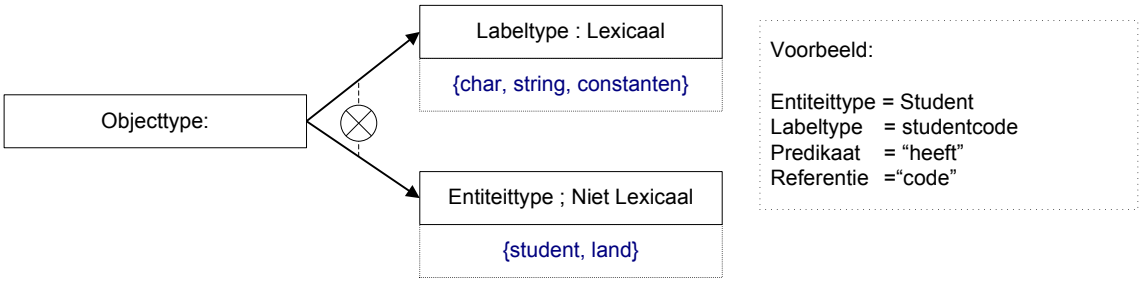
Om beter te kunnen begrijpen wat metamodellen zijn, is er in Figuur 5-9 een voorbeeld metamodel van de modelleertaal ORM te vinden. Dit model is een *mogelijk* metamodel voor ORM.



Elke subtype is een Objecttype, die een subtype is van een Objecttype
Elke labeltype is een Objecttype van het soort OT 'VT'.
Elke entiteittype is een Objecttype van het soort OT 'ET'.
Elke geobjectificeerd entiteittype is een entiteittype dat een relatie bouwt
Elke niet geobjectificeerde entiteittype is een entiteittype dat geen geobjectificeerd entiteittype is

Figuur 5-9 Meta(data) model van ORM [12]

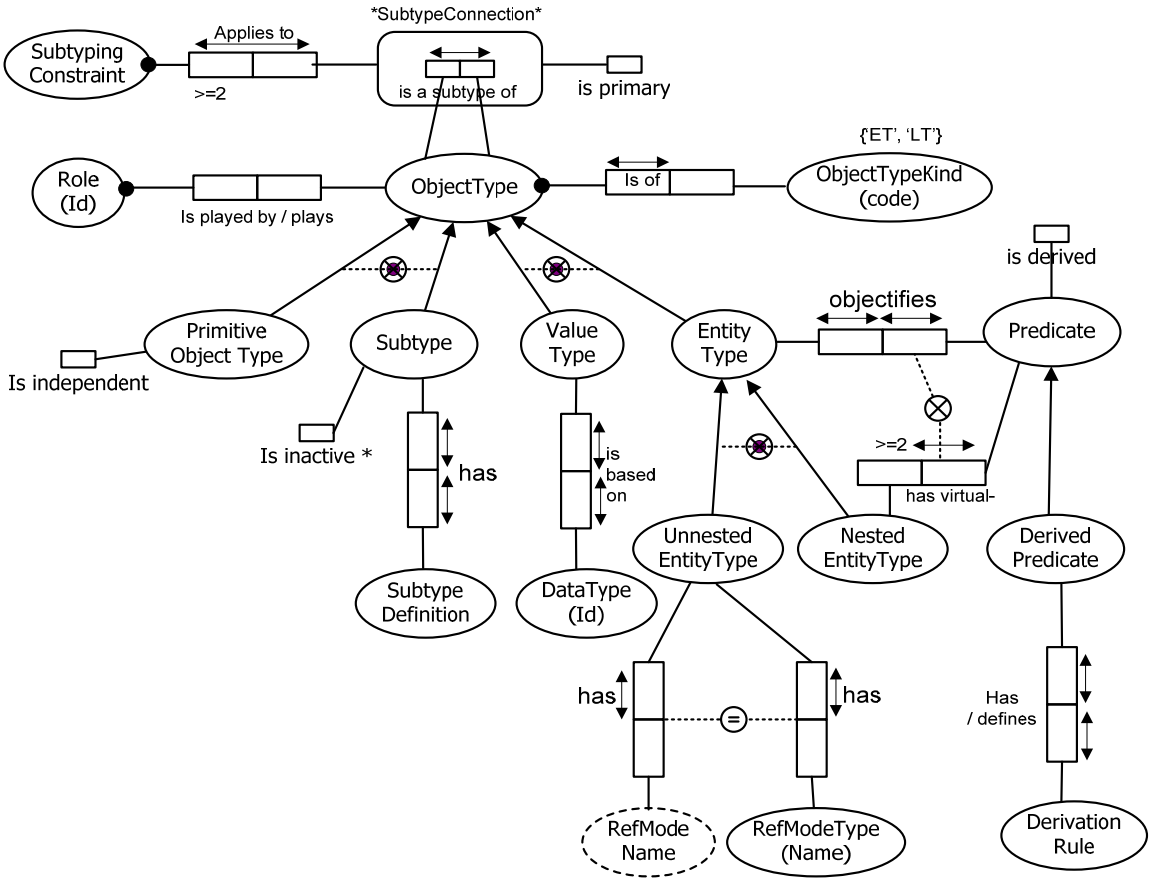
De twee voornaamste constructies in ORM zijn objecttypen en relaties. Een manier om deze te classificeren is weergegeven in Figuur 5-9, maar het kan ook anders. Labeltypen en entiteittypen worden ook lexicaal en niet lexicale objecttypen genoemd. Labels zijn constanten die een standaard denotatie hebben waardoor er geen referentie nodig is. Entiteiten, zoals studenten, en landen, hebben wel een referentie mode nodig. Een voorbeeld is een student met een studentcode '1234'. Bij een dergelijk geval wordt een objecttype aangeduid met een referentie zoals Student(code). De referentie mode is een manier waarop de waarde refereert naar de entiteit. Het relatietype van het voorbeeld Student is; "Student heeft studentcode". In bovenstaand metamodel wordt het relatietype aangegeven door "Relatie". Deze relatie refereert zowel naar het objecttype als het predikaat (bijvoorbeeld: ..heeft..). Dus "Student" is een entiteittype, "studentcode" is een labeltype, "heeft" is een predikaat en "code" is een referentie mode.



Figuur 5-10 Soorten objecttypen

Figuur 5-10 geeft aan dat een objecttype onderverdeeld kan worden in een labeltype (lexicaal) en een entiteittype (niet lexicaal). Het rondje met een x is een beperking (constraint), die aangeeft dat als het objecttype een labeltype is, dat het dan geen entiteittype kan zijn en omgekeerd ook niet. Een labeltype is bijvoorbeeld een constante / string en een entiteittype is bijvoorbeeld een student, land, werknemer etc.

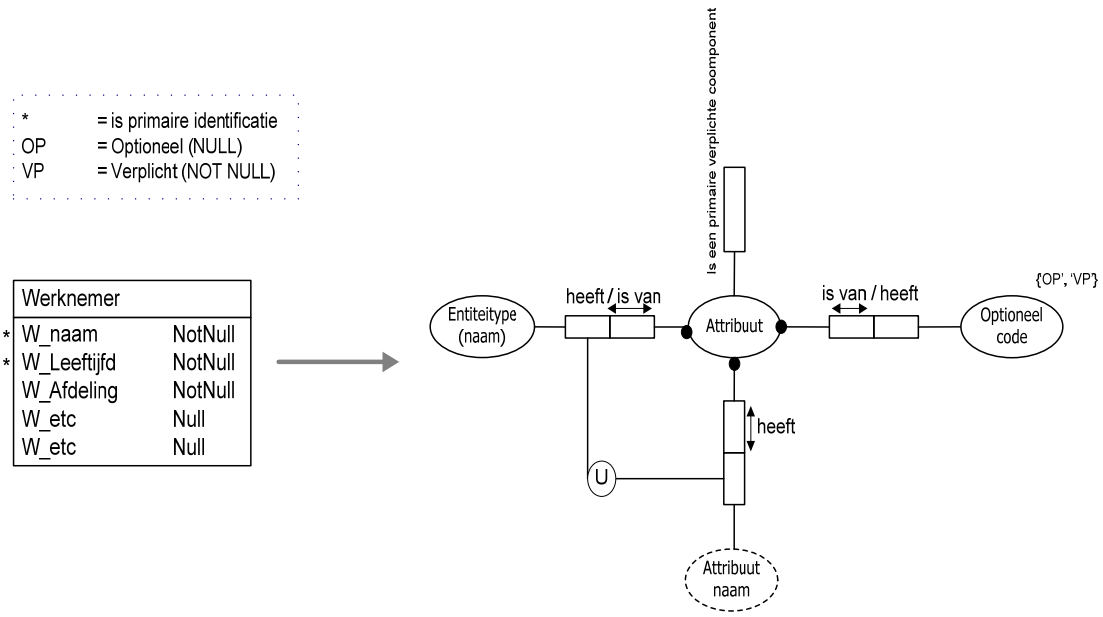
Figuur 5-9 is een mogelijk metamodel voor de modelleringstaal ORM. Dit betekent dat er ook andere metamodellen van ORM bestaan. Om dit verschil beter aan te geven is er in Figuur 5-11 een ander mogelijk metamodel gegeven van ORM.



Figuur 5-11 Metamodel 2 van ORM[30]

Ook in Figuur 5-11 is er een onderscheidt gemaakt tussen entiteittype (entity type)en labeltype (value type). Ook het subtype komt in dit metamodel weer voor. Bij beide metamodellen (Figuur 5-9 en Figuur 5-11) wordt aangegeven dat (alleen) een entiteittype geobjectificeerd (genest) kan worden. Bij Figuur 5-11 is ervoor gekozen om de definitie/omschrijving /predikaat bij het entiteittype aan te geven en niet het objecttype. Dit is niet fout, maar een andere manier om het te modelleren. De grootste verschillen in de twee metamodellen is dat er bij Figuur 5-11 ook de rol en “rol id” is gemodelleerd en het subtype connection.

De metamodellen in Figuur 5-9 en Figuur 5-11 zien er niet hetzelfde uit. Dit heeft te maken met de interpretatie, perceptie en voorkeur van een modelleur. Maar geen van beide metamodellen is fout. Het metamodel weergegeven in Figuur 5-9 geeft alle basis elementen van ORM weer. Indien een modelleur gebruik wil maken van meerdere mogelijkheden, beperkingen (constraints) in het model, dan is dit ook mogelijk. Alleen in dit geval zal het metamodel uitgebreid moeten worden met de extra mogelijkheden.

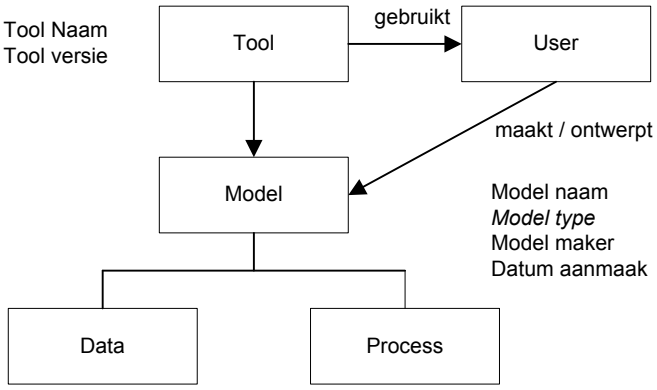


Figuur 5-12 Metamodel van tabel

In Figuur 5-12 is er een metamodel gemaakt van een tabel met gegevens van een werknemer, gebaseerd op [13]. Deze tabel heeft vijf kolommen waar gegevens van een werknemer opgeslagen kan worden. De eerste twee kolommen zijn de primaire identificatie van een werknemer, (ervan uitgaande dat de combinatie van leeftijd en naam uniek kan zijn voor de identificatie van een werknemer). Tevens wordt er per kolom aangegeven of een veld / gegeven van een werknemer verplicht is of niet (NULL en NOT NULL).

De gegevens van de werknemers tabel kunnen vervolgens in het metamodel worden vast gelegd. De werknemer, een entiteit, heeft attributen. Een attribuut heeft een naam en sommige attributen zijn verplicht en sommige zijn optioneel. Sommige attributen zijn primaire verplichte componenten.

Figuur 5-12 is een voorbeeld van een metamodel die los staat van een model. Het is een metamodel van een tabel. Om nog een voorbeeld hiervan te geven is er een metamodel gemaakt van het gebruik van tools om modellen te maken. Dit voorbeeld is te zien in Figuur 5-13.



Figuur 5-13 Een specifiek metamodel

Figuur 5-13 is een metamodel van tools die gebruikt worden door mensen (users) om modellen te maken [31]. Een tool heeft een naam en een versie, dit zijn meta gegevens. Een tool wordt gebruikt door een persoon, een gebruiker (user). De gebruiker maakt een model met behulp van de tool. Het model naam, type, maker en datum aanmaak zijn ook metagegevens van het model.

Dit metamodel is gebaseerd op een domein waarin gebruikers modellen maken met behulp van een tool. Tevens zijn er gewoon blokjes gebruikt om het metamodel te maken en niet een specifieke modelleertaal zoals bij Figuur 5-12.

5.8 OMG

OMG staat voor Object Management Group en is een consortium dat zich richt op het zetten van standaarden voor object georiënteerde systemen. OMG was opgericht in 1989 door elf bedrijven waaronder Hewlett Packard, Apple Computer Inc. en American Airlines. OMG had als doel bij de oprichting om een standaard architectuur voor gedistribueerde objecten in netwerken te creëren. In 1991 resulteerde dit idee voor een standaard architectuur in de Common Object Request Broker Architecture (CORBA). OMG heeft ook de standaard voor UML gecreëerd en gerelateerde technologieën als Meta Object Facility (MOF).

Het doel is om een framework aan te bieden dat alle soorten metadata kan ondersteunen. Om dit te kunnen bereiken gebruikt MOF een lagen metadata architectuur die gebaseerd is op het traditionele vier lagen meta modelling architecture van OMG MOF 1.X standaard. De belangrijkste eigenschap van deze architectuur is een meta meta modelling laag die een algemene taal biedt die het verband aangeeft tussen metamodellen en modellen. Het MOF model dat overeenkomt met het meta-metamodel in het traditionele vier lagen meta modelling architectuur is een object modelleringstaal dat nauw gerelateerd is aan UML. Het MOF model wordt gebruikt om de structuur en semantiek van een algemeen of domein specifiek metamodel te definiëren [23].

Het idee achter het MOF model is om meta-metamodellen te creëren in een standaard taal, MOF taal. Van elke taal, programmeertaal of modelleertaal, kan er een metamodel gemaakt worden. Deze metamodellen hebben verder niks met elkaar gemeen, behalve dat

ze allemaal metamodellen zijn. Het doel van MOF is om van al deze metamodellen een meta-metamodel te maken. Een meta-metamodel heeft het voordeel dat verschillende modellen met elkaar in relatie kunnen worden gebracht of vergeleken.

De Meta-Object Facility (MOF) van OMG vormt een brug voor het gat tussen ongelijke metamodellen. Dit kan mogelijk worden gemaakt door een basis te voorzien voor metamodellen. Als twee verschillende metamodellen MOF conform zijn, dan kunnen modellen die hierop gebaseerd zijn in dezelfde repository berusten. Een MOF metamodel definieert nieuwe constructoren corresponderend aan elementen van een modelleertaal of schema door een samenvoeging / verzameling van MOF constructen.

Zo is UML officieel gedefinieerd bij OMG door het UML metamodel. Dit metamodel is een MOF metamodel. Met andere woorden, dit metamodel is gebaseerd op MOF (conform MOF) en is een instantie van het meta-metamodel MOF.

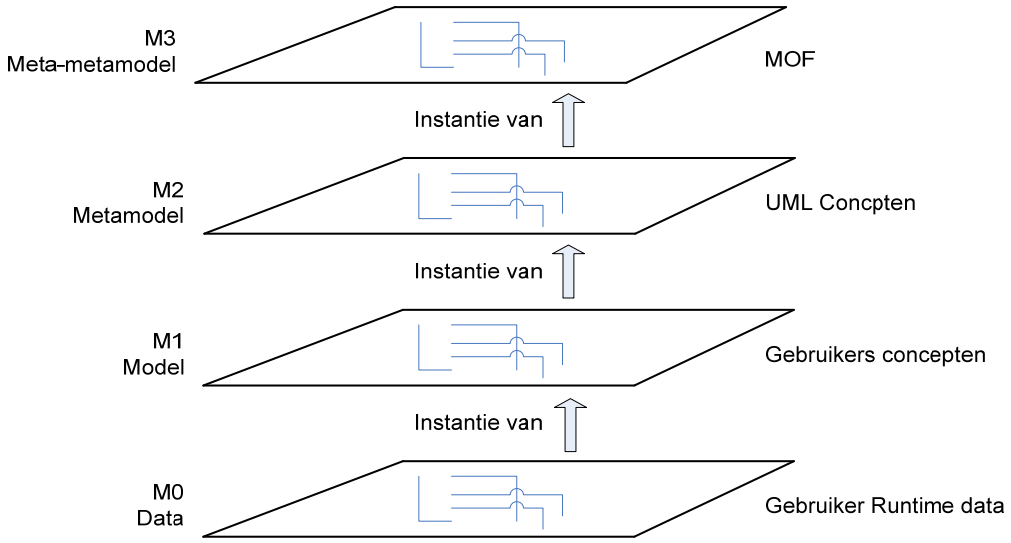
5.8.1 Vier lagen metamodel architectuur van OMG:

OMG hanteert een vier lagen metamodel. Dit is het metamodel architectuur. Deze architectuur is een bewezen infrastructuur voor het definiëren van (precieze) semantiek die nodig is voor complexe modellen. Hieronder volgen de vier lagen met een beschrijving ervan.

M0	Bevat de data van de applicatie. Bijvoorbeeld: de rijen in een relationele database tabel.
M1	Bevat de applicatie; de klassen van een object georiënteerde systeem of de tabel definitie van een relationele database. Application modelling
M2	Bevat het metamodel van de taal, bijvoorbeeld UML elementen zoals klassen, attributen.
M3	Het meta-metamodel dat de eigenschappen van alle metamodellen beschrijft.

De meta-metamodellering laag vormt de basis voor de metamodellering architectuur. De primaire verantwoordelijkheid van deze laag is om de taal voor het specificeren van een metamodel te definiëren. Een meta-metamodel definieert een model op een hoger abstractie niveau dan het metamodel. Tevens is het ook compacter dan het metamodel dat het beschrijft. Een meta-metamodel kan verschillende metamodellen definiëren en er kunnen meerdere meta-metamodellen geassocieerd zijn met elk metamodel. Hoewel het gewenst is dat gerelateerde metamodellen en meta-metamodellen dezelfde ontwerp filosofieën en constructen delen, is dit geen strikte regel. Elke laag heeft een eigen ontwerp integriteit. Voorbeelden van meta-meta-objecten in het meta-metamodel laag zijn: Meta-klassen, meta-attributen en meta-operaties [46].

In Figuur 5-14 wordt het vier lagen metamodel aangegeven. In dit figuur wordt de relatie aangegeven tussen de vier lagen. De eerste laag, (M0) bevat de gebruikersdata en hiervan kan een model gemaakt worden (M1). Van het model kan een metamodel (M2) gemaakt worden en van het metamodel kan weer een meta metamodel (M3) gemaakt worden (MOF).



Figuur 5-14 OMG vier lagen metamodel [14]

In Figuur 5-15 wordt het vier lagen model in tabel vorm aangegeven met een beschrijving en voorbeelden.

Layer	Description	Example
Meta-metamodel	De infrastructuur voor een metamodelleringsArchitectuur. Definieert de taal voor het specificeren van metamodellen	Metaklasse, Metaattribuut, Metaoperatie
metamodel	Een instantie van een meta-metamodel. Definieert de taal voor het specificeren van een model	Klasse, Attribuut, Operatie, Component
model	Een instantie van een metamodel. Definieert de taal om een informatie domein te beschrijven.	Stockshare, askPrice, sellLimitOrder, StockQuoteServer
Gebruikers objecten (data)	Een instantie van een model. Definieert een specifiek informatie domein.	<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

Figuur 5-15 OMG 4 layer hierarchy [14]

In Figuur 5-14 en Figuur 5-15 is de relatie tussen de vier lagen aangegeven. Een modelleertaal bestaat uit instanties van concepten in het metamodel. “Is instantie van” is de belangrijkste relatie tussen de verschillende lagen. Een UML metamodel is bijvoorbeeld een instantie van een MOF meta metamodel, dit wordt ook wel een loose metamodelling genoemd. Met andere woorden een Mn model is een instantie van een

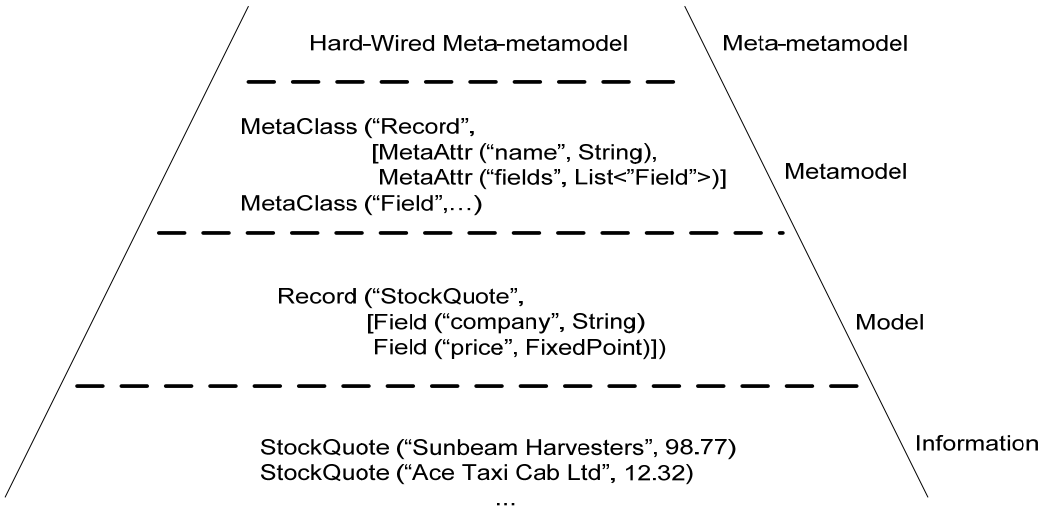
Mn+1 laag model. 1) In een n-laag architectuur model M0, M1 … Mn-1, zijn alle elementen van een Mm laag model een instantie van exact één element van een Mm+1 laag model, en voor alle m<n-1 en 2) elke relatie anders dan de "instantie van" relatie tussen twee elementen X en Y suggereert die level(X)=level(Y).

In formele notatie:

$\forall x \in Mm\ \exists y \in Mm+1(IV(x,y) \wedge \forall z \in Mm+1(IV(x,z) \rightarrow z=y))$.

IV(x,y) = x is instantie van y.

Voor alle x geldt, er is precies één y waarvan x een instantie is van y. En als er een andere z is waarbij x instantie is van z, dan is z gelijk aan y.



Figuur 5-16 OMG 4 layer hierarchy [14].

Het metamodel van OMG bestaat uit vier lagen. Als de vier lagen goed gedefinieerd zijn, dan is uitwisseling van modellen en van metamodellen mogelijk tussen producten die dezelfde meta-metamodellen hebben [1].

Ook al is de metadata niet gelijk, toch is het mappen van metadata van verschillende producten in principe mogelijk. Een mapping is een set regels en technieken om een model te kunnen omzetten naar een ander model.

5.9 Het metamodel hiërarchie: een framework voor informatiesystemen.

Naast het vier lagen metamodel architectuur zoals beschreven in de voorgaande paragraaf, is er ook een andere hiërarchie te onderkennen. Dit is het metamodel hiërarchie voor informatiesystemen [22]. Dit metamodel hiërarchie is niet hetzelfde concept als die van OMG en zal hier in deze paragraaf verder worden uitgelegd.

Het metamodel hiërarchie is een framework voor het begrijpen en vergelijken van modelleringstechnieken. Verschillende onderzoekers creëren nieuwe modelleertalen, maar de reden en de waarde voor de nieuwe taal of techniek wordt niet geadresseerd. Hierdoor bestaan er tegenwoordig verschillende modelleringstechnieken met verschillende modelleertalen, die misschien geen extra waarde hebben en dus overbodig zijn in het vak. Tevens verschillen de technieken allemaal onderling van elkaar. Dit brengt een aantal problemen met zich mee. Veel energie en mentale kracht worden verspild waneer;

1. Er voor een nieuwe applicatie een geschikte modelleringstechniek geselecteerd moet worden;
2. Er bij re-engenireering van bestaande applicatietranslaties tussen modelleringstechnieken moet worden vastgesteld;
3. Of wanneer docenten aan de studenten moeten leren over modelleringstechnieken.

Het idee van het metamodel hiërarchie is ontworpen om bovenstaande problemen op te lossen.

5.9.1 Ordenen van metamodellen

Het metamodel hiërarchie is gebaseerd op de relaties en corresponderende operatoren en metamodellen. Er worden drie relaties van metamodellen gedefinieerd, namelijk 1. partitioning (Part), 2. restriction (Restr) en 3. degenreation (Deg). De relaties vormen samen een gerichte acyclische graaf (waarbij een knoop is zonder inkomende takken) tussen de metamodellen. Deze gerichte acyclische graaf tussen metamodellen vormt het metamodel hiërarchie.

Formele definitie metamodel hiërarchie [22]:

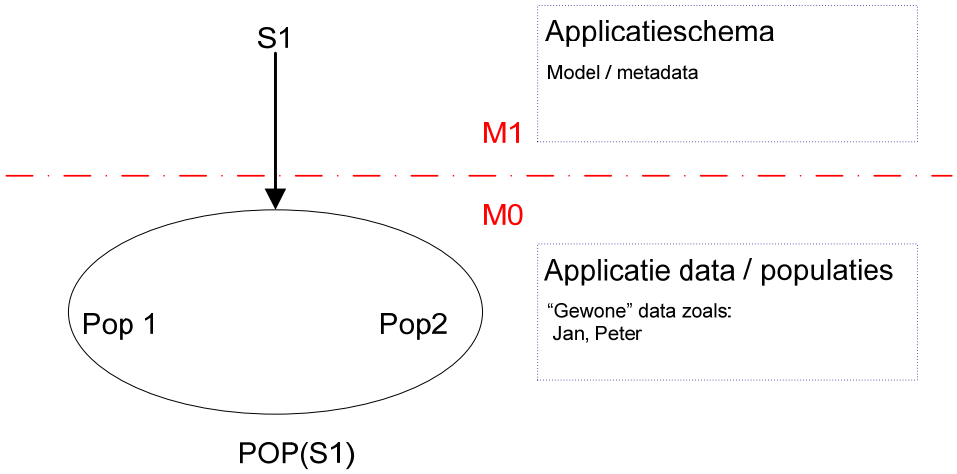
> *The meta model Hierarchy MMH, which is a directed acyclic graph, has the following structure:*
>
> *MMH= <**MM**, <MMH, Part,Restr,Deg>*
> *Where MM is the set of possible metamodels,*
> *Part ⊆ **MM** x **MM** is a relation on meta models, capturing partitioning,*
> *Restr ⊆ **MM** x **MM** is a relation on meta models, capturing restriction,*
> *Deg ⊆ **MM** x **MM** is a relation on meta models, capturing degeneration,*
> *<MMH ⊆ **MM** x **MM** is a relation on meta models, defined as follows:*
> *X < MMH y iff (xParty) or (x Restr y) or (x Deg y).*

Het idee achter dit metamodel hiërarchie is om verschillende metamodellen te kunnen vergelijken met elkaar en mappen tussen verschillende modellen. Dit metamodel hiërarchie gaat uit van een hogere top, genoemd de root voor het kunnen vergelijken van deze metamodellen. OMG gaat er ook uit van een hogere top (MOF) om metamodellen te

vergelijken of mappen, maar deze twee concepten zijn gebaseerd op twee verschillende theorieën en dienen gezien te worden als twee verschillende concepten.

Voor meer informatie over het metamodel hiërarchie, restriction, partitioning en degeneration wordt verwezen naar [22]

5.10 Schema populatie tweedeling



Figuur 5-17 POP(S1)

Zoals eerder uitgelegd kan van de data / applicatiedata een schema worden gemaakt. Als S een schema is, dan is POP(S) de toestandsruimte van S. De toestandsruimte POP(S) is de verzameling van alle mogelijke populaties van S. In Figuur 5-17 wordt een voorbeeld gegeven van een schema S1 met de populaties weergegeven als pop1 en pop2. Hierbij is de toestandsruimte POP(S1) de verzameling van alle mogelijke populaties van S1. De applicatiedata Pop1 en Pop2 behoren tot de M0 niveau van de OMG laag. Het applicatieschema S1 behoort tot de M1 niveau van het OMG vier lagen model.

5.10.1 Schema's en populaties

We maken data modellen. Elk data model heeft een schema dat uit feittypen bestaat.

Twee voorbeelden van schema's, (applicatieschema's):

S1 = {f,g} waarbij f = [A,B]
g = [B,C]

S2 = {f,h} waarbij f = [A,B]
h = [B,D]

Bijvoorbeeld stel schema S1 = {f,g}. Laat Pop1 en Pop2 twee verschillende populaties zijn van S. Dus Pop1, Pop2 $\in$ POP(S1) $\wedge$ Pop1 $\neq$ Pop 2.

Pop1

A	B
a1	b1
a1	b2

B	C
b1	c1

Pop2

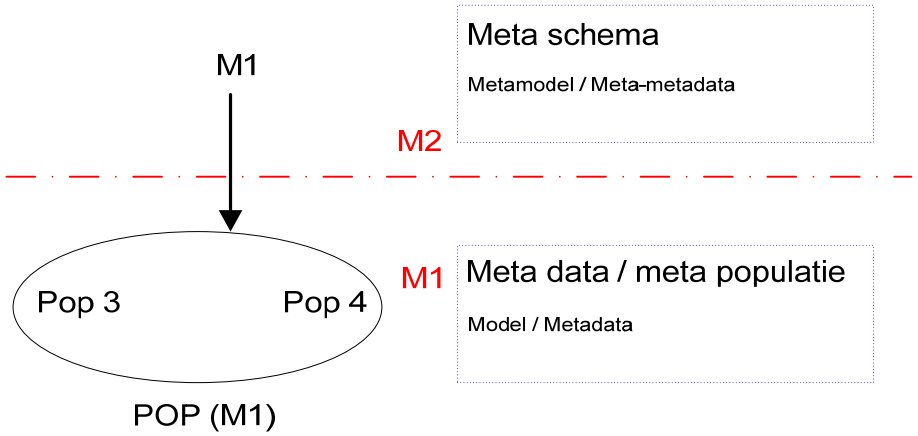
A	B
a1	b1
a1	b2
a2	b3

B	C
b1	c1
b3	c1

Figuur 5-18 Voorbeeld Pop1 en Pop2

In Figuur 5-18 wordt aangegeven hoe Pop1 en Pop2 eruit zouden kunnen zien.

5.10.2 Meta-Schema Metapopulatie Tweedeling



Figuur 5-19 Pop(M1)

Zoals eerder uitgelegd, kan er van een applicatieschema een metaschema gemaakt worden. Het applicatieschema is hierbij een instantie van het metaschema. Een metaschema heeft hierdoor ook populaties. Een populatie van een metaschema beschrijft een applicatieschema. In Figuur 5-19 is dit weergegeven. Ook hier is te zien dat Pop 3 en Pop4 tot de M1 niveau van de OMG laag behoren. Het metamodel behoort tot de M2 laag.

5.10.3 Metaschema en populaties

Een metaschema bestaat ook uit populaties net zoals een model uit populaties bestaat. In dit geval beschrijft de populatie van het metaschema, het applicatieschema.

Bijvoorbeeld meta schema M1 = {F} waarbij F = [feittype, rol]

POP(M1) is de toestandsruimte van M1. Pop(M1) bevat alle populaties van M1. Met andere woorden De toestandsruimte POP(M1) is de verzameling van alle mogelijke populaties van M1. Dit is weergegeven in Figuur 5-19.

Laat Pop 3 en Pop 4 twee verschillende populaties zijn van M1. Dus Pop3 en Pop4 $\in$ POP(M1) $\wedge$ Pop 3 $\neq$ Pop 4.

Nu stellen we: Pop3 beschrijft S1.
Pop4 beschrijft S2.

S1 en S2 zijn beide applicatieschema's die weer op hun beurt populaties bevatten. In Figuur 5-17 is applicatieschema S1 te zien met populaties. Mogelijke populatie voorbeelden van Pop 3 en Pop 4 is te zien in Figuur 5-20.

Pop3

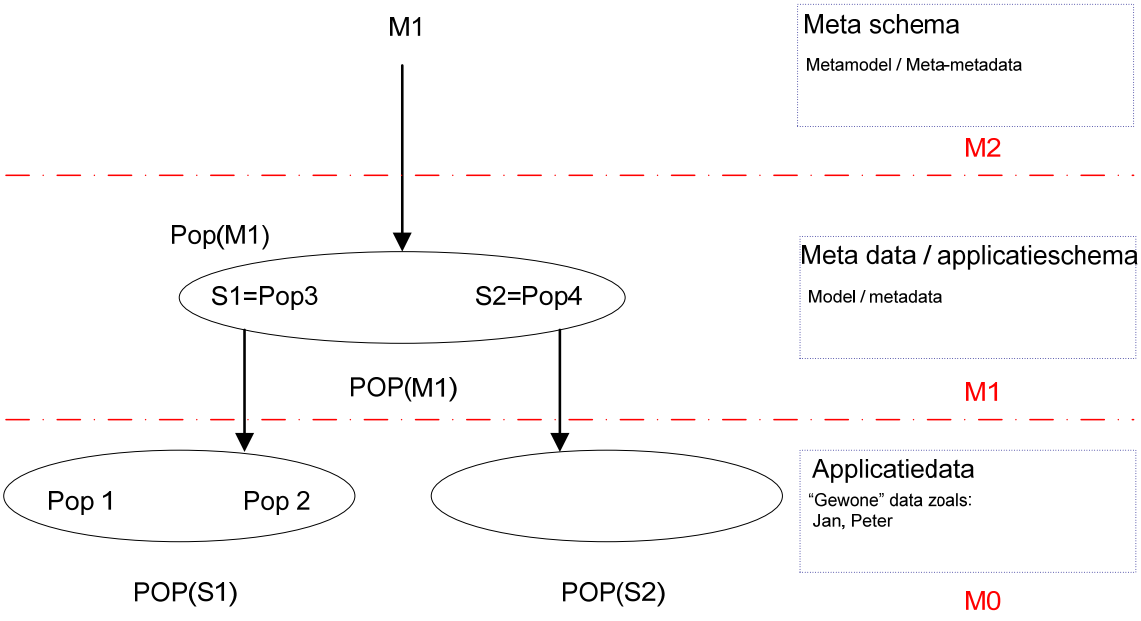
feittype	rol
f	A
f	B
g	B
g	C

Pop4

feittype	rol
f	A
f	B
h	B
h	D

Figuur 5-20 Voorbeeld Pop3 en Pop4

5.10.4 Samenvoeging van de Tweedelingen



Figuur 5-21 Samenvoeging van de tweedelingen

Het gehele verhaal van metaschema's, applicatieschema's en applicatie data is weergegeven in Figuur 5-21. Hierin wordt de samenhang tussen de verschillende modellen aangegeven. In Figuur 5-21 wordt aangegeven hoe een metaschema M1 uit meta-metadata bestaat. Deze meta-metadata zijn de populaties van de applicatieschema's S1 en S2. S1 en S2 bevatten vervolgens weer de populaties Pop1 en Pop2 enz.

Dit roept allerlei interessante vragen op. Bijvoorbeeld: kunnen verschillende schema's dezelfde populatie hebben? Dus in dit geval, kunnen POP(S1) en POP(S2) elkaar overlappen?

In hoofdstuk 6 wordt verder uitgelegd hoe de relatie tussen data, metadata, modellen en metamodellen eruit ziet.

6. Metadata en metamodel

In de voorgaande hoofdstukken is een beschrijving gegeven van wat metadata en metamodellen zijn met behulp van een aantal praktische voorbeelden. De volgende stap is de relatie/samenhang tussen metadata en metamodellen aan te geven. Dit zal worden gedaan met behulp van het OMG vier lagen model.

De grootste verschillen tussen metadata en metamodel zijn:

- Metadata zijn gestructureerde data over data of met andere woorden, het zijn de eigenschappen van objecten.

- Een metamodel wordt simpel vertaald als een model van een model, maar dan met een hogere abstractie niveau.

Data zijn gewoon gegevens en een model is gebaseerd op gegevens. Data is gemodelleerd door metadata (schema's, classes etc). Deze vormen allemaal een gedeelte van het metamodel. Met andere woorden, ze zijn voorbeelden van metadata die op hun beurt weer kunnen behoren tot het metamodel. Maar data kan ook in een model worden gemodelleerd. Dus metadata is ook een model.

6.1 Relatie tussen het metamodel, metadata en data

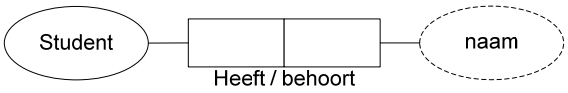
In hoofdstuk twee is uitgelegd dat metadata toegepast wordt in de verschillende IT werelden. Eigenlijk hebben alle objecten eigenschappen en is er dus sprake van metadata. Zelfs in een relationele database is er metadata aanwezig die de catalogus wordt genoemd. Maar een gebruiker hoeft niet de catalogus van een database te benaderen om metadata te kunnen bekijken, metadata is ook gewoon in een tabel van een database aanwezig. Figuur 6-1 is een voorbeeld van een tabel waarin student gegevens worden opgeslagen, (De data die te vinden is in de tabel, zoals jan, peter, straat 1 enz behoren tot het m0 niveau van OMG). Deze tabel heeft een aantal eigenschappen, zoals studentnummer, studentnaam, adres, woonplaats en telefoonnummer. Dit zijn eigenlijk eigenschappen van de entiteit "STUDENT". In een model zoals UML en ORM worden ook objecten (entiteiten) en hun eigenschappen (onderlinge relaties) gemodelleerd. Van onderstaande tabel kan er dus een model gemaakt worden.

TABEL: STUDENT

STUDENTNR	STUDENTNAAM	ADRES	WOONPLAATS	TELEFOONNR
1	Jan	Straat 1	Zutphen	1234
2	Peter	Straat 2	Arnhem	5678
3	Klaas	Straat 3	Nijmegen	9101
4	Willem	Straat 4	Ede	1213

Figuur 6-1 tabel student

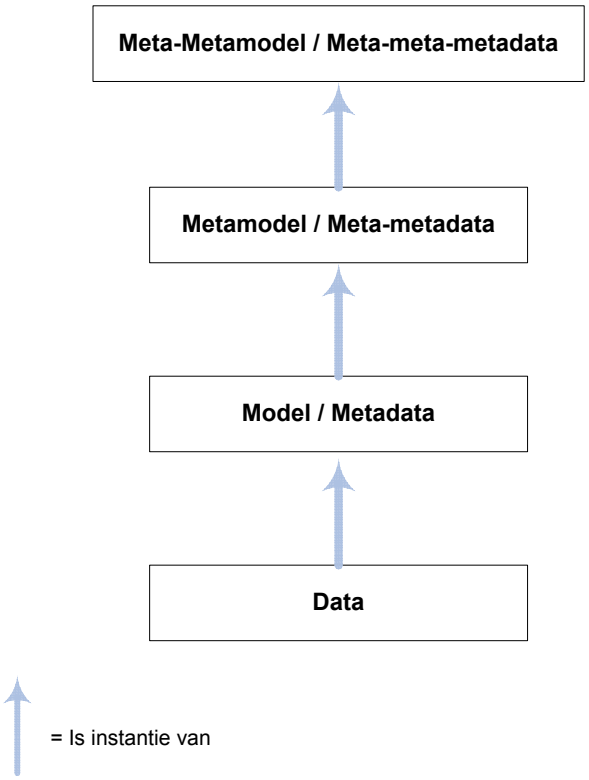
Om een voorbeeld model te maken van Figuur 6-1 zal er gebruik worden gemaakt van het volgende gegeven (feittype): "Een student heeft een naam (studentnaam)". Figuur 6-2 geeft het model van dit feittype weer.



Figuur 6-2 Een voorbeeld model van tabel student

6.1.1 Hiërarchische structuur

Eerder is aangegeven dat de eigenschappen van entiteit student eigenlijk metadata zijn. Dit betekent dat er in Figuur 6-2 een entiteit met metadata is gemodelleerd. Met andere woorden; in een model worden objecten met hun eigenschappen (metadata) gemodelleerd. Een model bevat metadata. Als er bij een model sprake is van metadata, dan kan er geconcludeerd worden dat een metamodel meta-metadata bevat en een meta-metamodel bevat meta-meta-metadata. Deze hiërarchische structuur tussen data, model, metadata en metamodel wordt in Figuur 6-3 aangegeven.



Figuur 6-3 Hiërarchische structuur tussen data, model, metadata en metamodel

In hoofdstuk 5 is uitgelegd dat een model uit instanties bestaat van concepten in het metamodel. Data zijn dus een instantie van model / metadata. Model / metadata is een instantie van metamodel / meta-metadata. Metamodel en meta-metadata zijn een instantie van meta-metamodel en meta-meta-metadata.

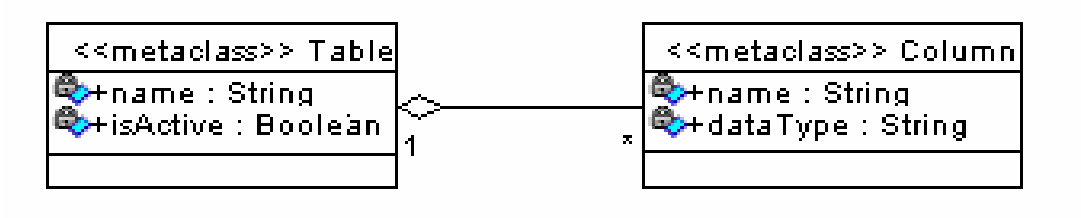
Om de relatie tussen data, model, metadata en metamodel beter uit te leggen zal de tabel STUDENT in Figuur 6-1 nogmaals gebruikt worden. Een entiteit zoals student heeft een aantal eigenschappen, dit zijn de metadata. Op een hoger niveau kan er gezegd worden dat een tabel (en niet de entiteit) in een database zelf ook een aantal eigenschappen heeft. Een tabel die kolommen bevat is een eigenschap van de tabel. De kolommen van een tabel kennen attributen, zoals kolom STUDENTNR = integer en STUDENTNAAM = string enz. Deze gegevens zijn ook metadata, omdat ze de tabel beschrijven.

TABEL: STUDENT

STUDENTNR	STUDENTNAAM	ADRES	WOONPLAATS	TELEFOONNR
1	Jan	Straat 1	Zutphen	1234
2	Peter	Straat 2	Arnhem	5678
3	Klaas	Straat 3	Nijmegen	9101
4	Willem	Straat 4	Ede	1213

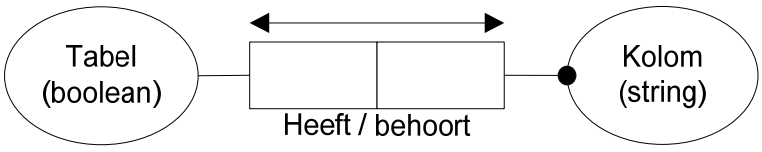
Figuur 6-1 tabel student

De tabel in Figuur 6-1 heeft kolommen, dus tabellen hebben kolommen. Deze eigenschap kan in een model worden weergegeven. Dit model is dan eigenlijk een metamodel, omdat het een model is van een model. Het metamodel van bovenstaand tabel is te zien in Figuur 6-4.



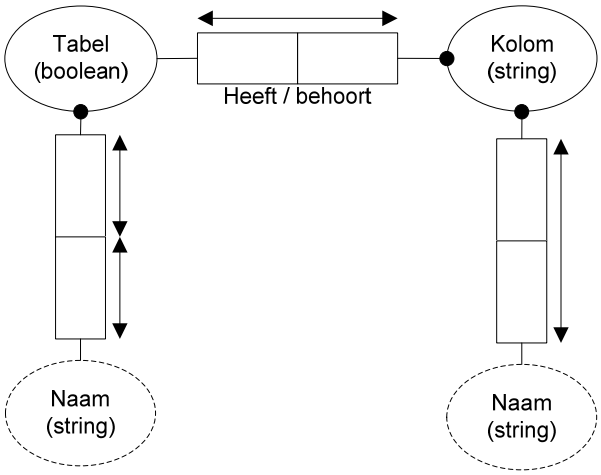
Figuur 6-4 UML metamodel van een tabel en kolommen [1]

Figuur 6-4 is een metamodel gemaakt in UML. Dit model legt vast dat iedere tabel meerdere kolommen kan bevatten, (dit is aangegeven door de asterisk *) en dat iedere kolom bij één tabel hoort (dit is aangegeven door de “1”). Om metamodellen goed te kunnen beschrijven zijn er definities nodig van de “dingen” die in de metamodellen worden gehanteerd: het meta-metamodel. Dan gaat het over “wat zijn relaties” en “alles heeft een naam” [1]. Van het metamodel zoals weergegeven in Figuur 6-4 kan ook een ORM versie gemaakt worden. Zie hiervoor Figuur 6-5.



Figuur 6-5 metamodel van een tabel en klommen

In Figuur 6-5 wordt aangeven dat een tabel kolommen bevat en dat een kolom bij een tabel behoort.



Figuur 6-6 Tweede versie ORM metamodel

Figuur 6-6 is een uitbreiding van het metamodel in Figuur 6-5. De kolommen en tabellen hebben beide een naam van het type "string". Een tabel (naam) kan niet vaker in een database voorkomen. Een kolom (naam) kan wel vaker in een tabel voorkomen. Dit dient om onder anderen de relatie aan te geven tussen de primaire en verwijzende sleutel.

6.2 Metamodel, metadata en OMG

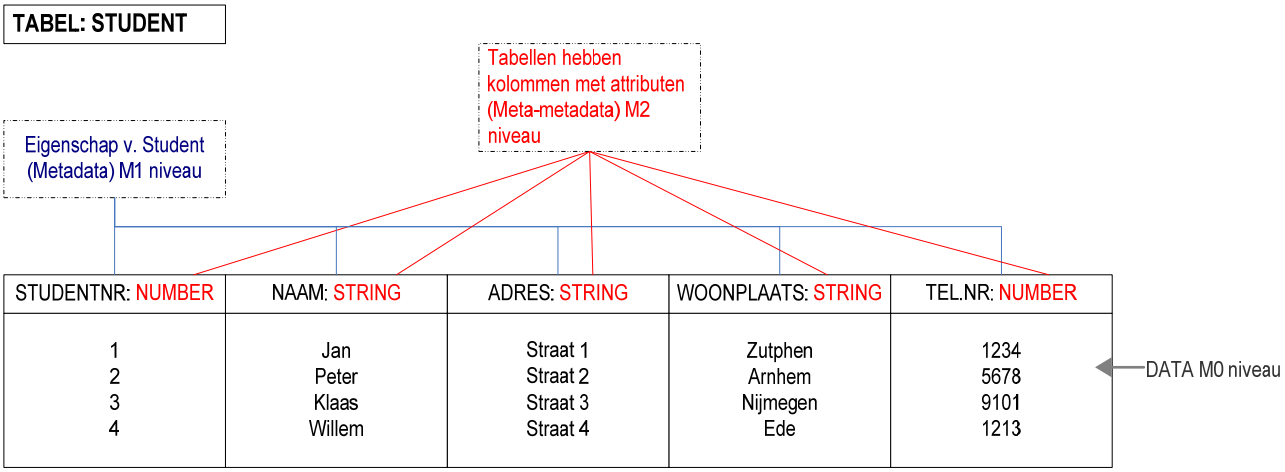
Het verband tussen, data, model, metadata en metamodellen is uitgelegd in de voorgaande paragraaf. De hiërarchische structuur tussen deze termen is weergegeven in Figuur 6-3. Om een compleet beeld te geven van de relatie tussen deze termen, zal de niveaus die OMG hanteert (M0, M1, M2 en M3, zie ook hoofdstuk 5) worden aangeven in Figuur 6-7.

Abstractie niveau	Voorbeeld	Naamgeving		Voorbeeld
M3	DINGEN kunnen RELATIES hebben	Meta-metamodel	Meta-meta-metadata	MOF
M2	TABELLEN hebben KOLOMMEN	Metamodel	Meta-metadata	UML/ORM
M1	STUDENTEN hebben een WOONPLAATS	Model	Metadata	Tabel
M0	JAN woont in ZUTPHEN	Data	Data	-

Figuur 6-7 OMG, metadata en metamodellen [1]

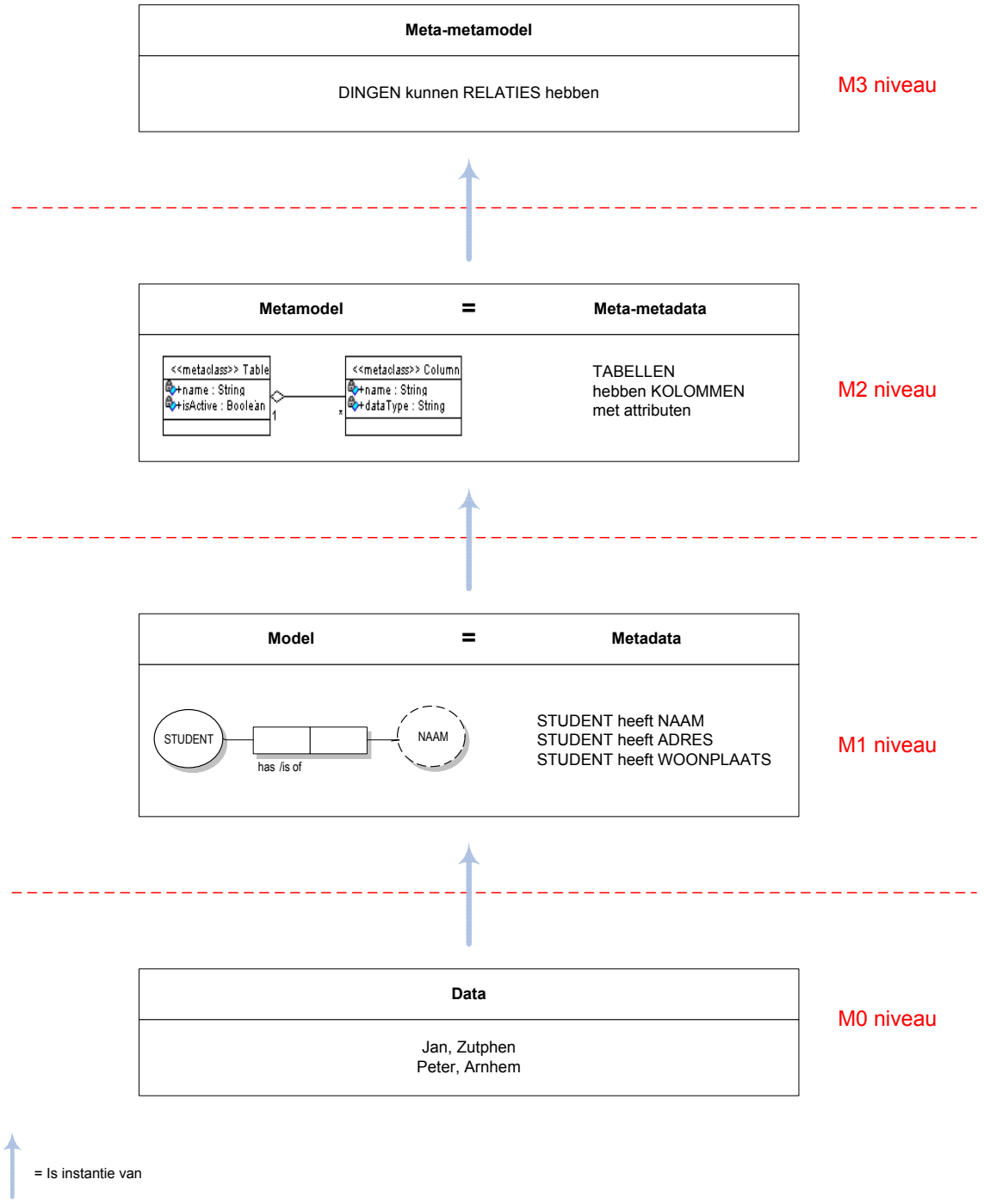
In Figuur 6-7 wordt de relatie van metadata en metamodellen weergegeven in het OMG vier lagen model. “Gewone” data kan een instantie zijn van zowel een model als van metadata.

Op M0 niveau bevindt zich de “gewone” data, zoals “Jan”, “Peter”, “Arnhem”, “Zuthpen” etc zie ook Figuur 6-8. M1 niveau beschrijft een bepaald gegeven op M0 niveau. In het voorbeeld van de tabel student is; “STUDENTEN hebben een WOONPLAATS”, een model of de metadata op M1 niveau. Het object STUDENT heeft aantal eigenschappen zoals, “NAAM”, “WOONPLAATS” etc. Deze eigenschappen zijn de metadata op M1 niveau. Op M2 niveau wordt het metamodel gemodelleerd. Een tabel heeft kolommen en deze kolommen hebben eigenschappen, het meta-metadata. De database objecten kunnen weer gezien worden als een instantie van een hoger abstractieniveau, het M3 niveau. Op het M3 niveau geldt dus dat “dingen” een naam en een aantal eigenschappen hebben, dat voor alle “dingen” hetzelfde zijn. Een voorbeeld van M3 niveau is MOF van OMG.



Figuur 6-8 Tabel student in combinatie met OMG

Model en metadata zijn dus op hetzelfde niveau, namelijk M1. Dit geldt ook voor meta-metadata en metamodel. Figuur 6-9 geeft een hiërarchische structuur van de relatie tussen de termen data, metadata, model en metamodel in combinatie met de OMG vier lagen niveaus. Tevens is er per OMG niveau ook een voorbeeld gegeven zodat er een beter beeld gecreëerd kan worden van het totale plaatje.



Figuur 6-9 Hiërarchische samenhang van metadata, metamodel en OMG

6.3 Abstract hiërarchisch model

De relatie tussen de termen data, model, metadata, meta-metadata en meta-metamodel is in de voorgaande paragrafen uitgelegd. Deze relatie is tevens in context gebracht met het vier lagen model van OMG, zie hier voor onder anderen Figuur 6-3 en Figuur 6-9. Deze figuren zijn echter nog niet compleet. In de voorgaande hoofdstukken is er een onderscheidt gemaakt tussen verschillende soorten metadata en verschillende soorten modellen en metamodellen. Deze onderverdelingen zijn niet meegenomen in Figuur 6-3 en Figuur 6-9.

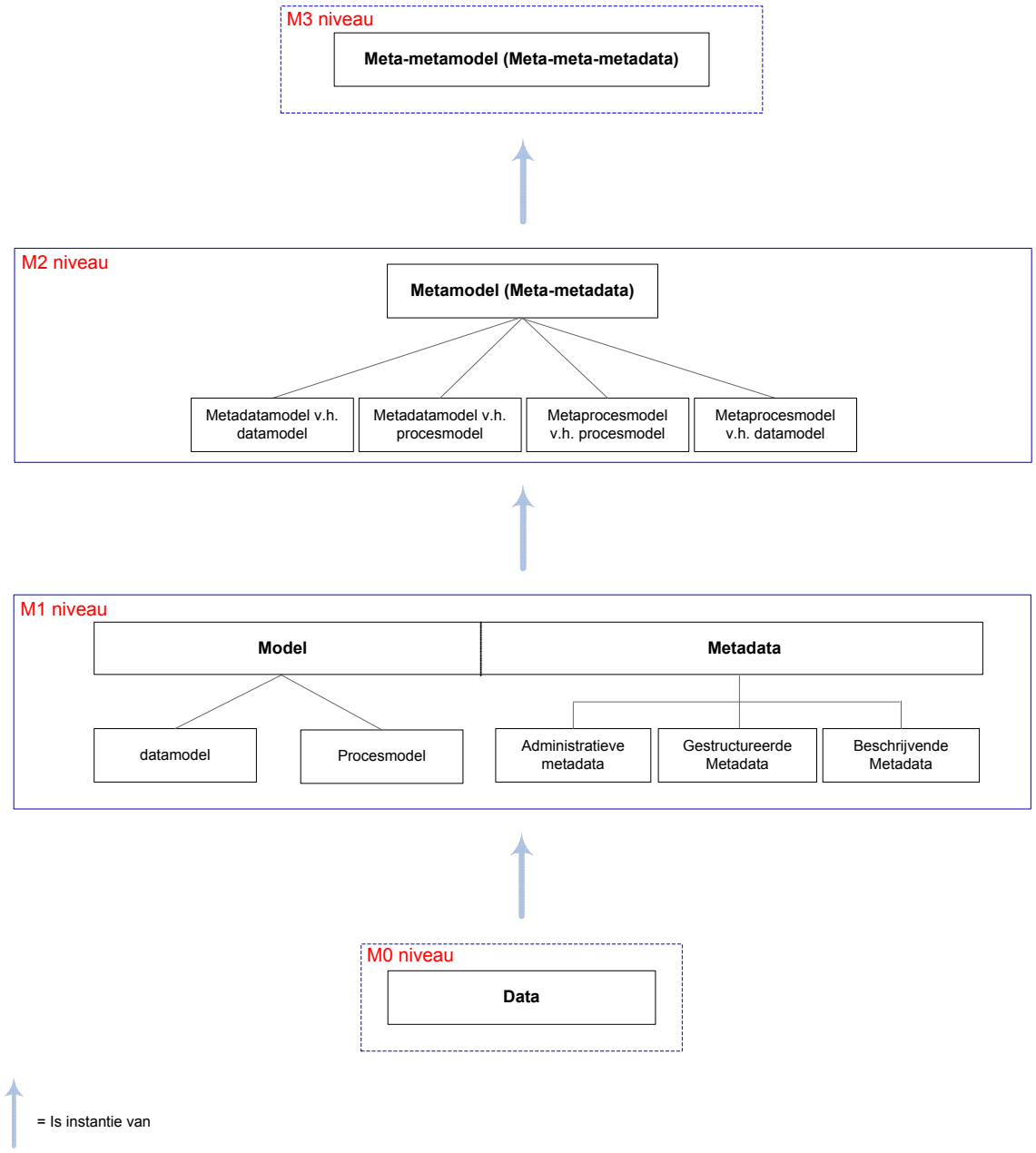
Om alle soorten metadata, modellen en metamodellen in kaart te brengen is er een abstract hiërarchisch model gemaakt, zie hiervoor Figuur 6-10. Dit model geeft alle onderverdelingen en soorten metadata en modellen weer in combinatie met het OMG vier lagen model.

Bij Figuur 6-10 is de principiële onderverdeling te zien tussen de 4 OMG niveaus (M1, M2, M3 en M4). Op M0 niveau is er geen onderscheidt gemaakt tussen verschillende soorten data (onder data vallen allerlei soorten gegevens die er zijn). De verschillende soorten metadata; administratieve-, gestructureerde- en beschrijvende metadata, zijn te vinden onder hoofdgroep “metadata” op M1 niveau. Ook bij modellen kan er een onderscheidt gemaakt worden tussen een datamodel en een procesmodel. Deze zijn te vinden onder hoofdgroep “model” op M1 niveau. De verschillende soorten metamodellen; 1) metadatamodel van het datamodel, 2) metadatamodel van het procesmodel, 3) metaprocesmodel van het procesmodel en 4) metaprocesmodel van het datamodel zijn onder hoofdgroep “Metamodel” te vinden op M2 niveau. M2 niveau bevat ook de meta-metadata gegevens. M3 niveau bevat het meta-metamodel en meta-meta-metadata. Bij deze hoofdgroepen is er geen onderverdeling gemaakt in soorten meta-metamodel of meta-meta-metadata.

Ook hier blijft gelden dat alle elementen van een Mm laag model een instantie zijn van exact één element van een Mm+1 laag model. Figuur 6-10 is een abstract model waar de termen data, metadata, model, metamodel en meta-metamodel in voorkomen.

6.4 Meta is hoger abstractie niveau?

De definitie van metadata zoals gehanteerd in deze scriptie luidt als volgt: “Metadata zijn de eigenschappen van een gegeven informatie zowel fysiek als elektronisch”. Voor een metamodel luidt de definitie als volgt: “metamodel is een model gemaakt op een hoger abstractie niveau die de regels en constructies (syntax en semantiek) van het onderliggende model aangeeft”. Met andere woorden een metamodel beschrijft een model, zoals een modelleertaal, op een hoger abstractie niveau. Als dit voor een metamodel geldt, dan geldt het ook voor metadata. Figuur 6-10 laat zien dat metadata op een hoger niveau is dan data. En meta-metadata is op een hoger niveau dan metadata. Elke Mm+1 niveau is abstracter dan het Mm niveau. Hierdoor geldt het dat elke Mm+1 niveau een hoger abstractie niveau heeft dan het Mm niveau.

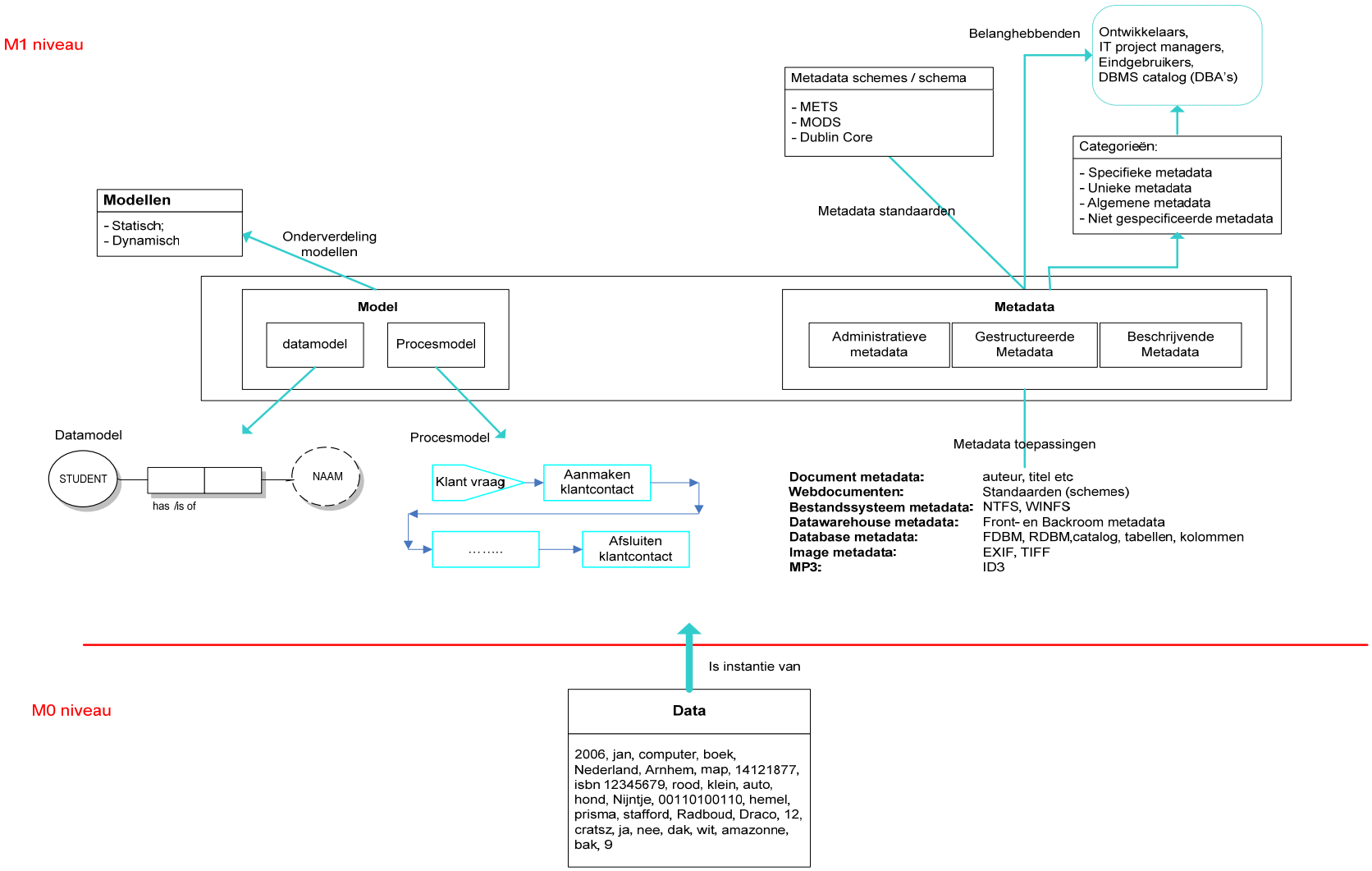


Figuur 6-10 Abstract (hiërarchisch) overkoepelend model

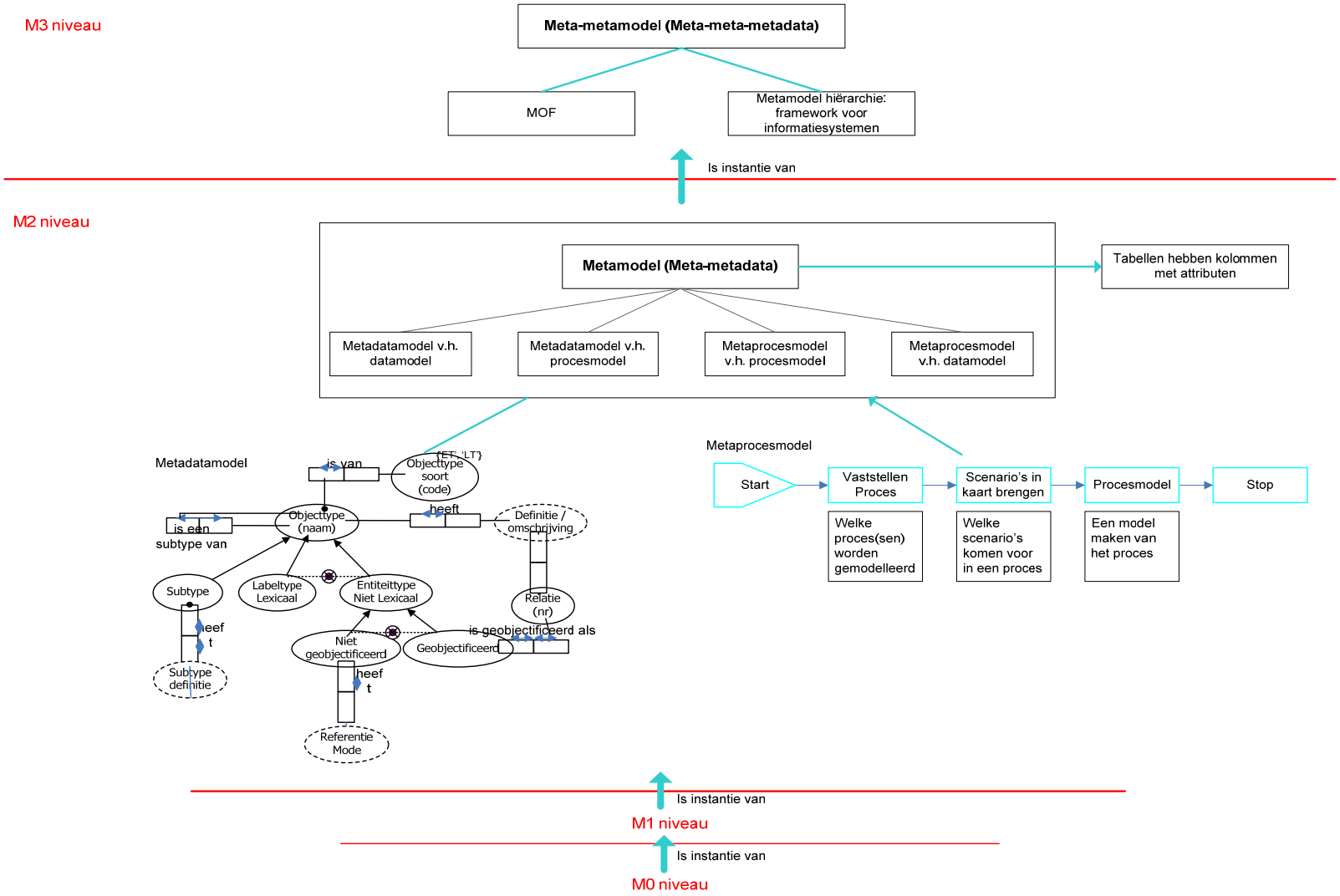
In Figuur 6-10 is de samenhang tussen data, metadata en metamodel weergegeven. Dit figuur kan nog extra uitgebreid worden met de voorbeelden die gegeven zijn in de voorgaande hoofdstukken. In hoofdstuk 3 wordt metadata uitgebeid uitgelegd met voorbeelden van toepassingen in de praktijk. Ook van metamodel is er in de hoofdstukken 4 en 5 voorbeelden gegeven. Deze voorbeelden van metadata en metamodel kunnen ook in het abstract (hiërarchisch) overkoepelend model geplaatst worden. Figuur 6-11 is een uitbreiding van het abstract (hiërarchisch) overkoepelend model met de voorbeelden die gegeven zijn in de voorgaande hoofdstukken.

Figuur 6-11 is verdeeld in figuur A en B, omdat het geheel niet op één pagina past. Figuur 6-11A is het (gedetailleerd hiërarchisch) overkoepelend model van niveau 0 en 1. Op niveau 0 bevindt zich de gewone data, zoals; “jan, Arnhem, 01-12-1990” en nog veel meer. Op m1 niveau bevindt zich de metadata en het model. Een model is in deze scriptie onderverdeeld in een dynamisch en een statisch model. Vervolgens zijn er de volgende soorten modellen; datamodel en procesmodel. Van elk soort model wordt een voorbeeld gegeven. Bij metadata wordt een onderverdeling gemaakt in de soorten; administratieve-, gestructureerde- en beschrijvende metadata. Ook de toepassingen van metadata in de praktijk zijn meegenomen in het plaatje. Deze zijn te vinden onder metadata. De metadata toepassingen zijn niet onderverdeeld in de soorten metadata, omdat de toepassingen niet verdeeld kunnen worden in één specifiek metadata soort. De categorieën van metadata worden ook aangegeven. Deze zijn; specifieke-, unieke-, algemene- en niet specifieke metadata. Ook de belanghebbenden zijn aangegeven. Als laatste wordt de metadata standaarden aangegeven. Deze zijn; METS, MODS en Dublin Core. In de praktijk zijn er meerdere standaarden, maar in deze scriptie zijn alleen deze drie behandeld.

Figuur 6-11B behandelt de niveaus 2 en 3. Op m2 niveau bevindt zich de metamodellen. De verschillende soorten metamodellen zijn; metadatamodel van het datamodel, metadatamodel van het procesmodel, metaprocesmodel van het datamodel en metaprocesmodel van het procesmodel. Er worden hier twee voorbeeld metamodellen gegeven, namelijk; een metadatamodel en een metaprocesmodel. Op m3 niveau bevindt zich het meta-metamodel. In deze scriptie zijn twee theorieën voor het meta-metamodel behandeld. Het eerste is het MOF van OMG en het tweede is het metamodel hiërarchie; een framework voor informatiesystemen. Beide theorieën gaan uit van een hogere top dan het metamodel, namelijk het meta-metamodel of root zoals die genoemd wordt bij het metamodel hiërarchie.



Figuur 6-11A Gedetailleerd (hiërarchisch) overkoepelend model



Figuur 6-11 B Gedetailleerd (hiërarchisch) overkoepelend model

7. Conclusie

In dit hoofdstuk zal er een conclusie gegeven worden van het onderzoekstraject. De onderzoeksvragen, hoofdvraag en een evaluatie van het gehele proces zullen aan bod komen.

De hoofdvraag in deze scriptie luidde: "*Hoe kunnen metadata en metamodellen als basis dienen voor het genereren van de gewenste informatie?* "

Om deze hoofdvraag te beantwoorden is er een onderverdeling gemaakt in een aantal onderzoeksvragen.

7.1 Beantwoording van de onderzoeksvragen

Gedurende dit onderzoek zijn de onderzoeksvragen beantwoordt. Hieronder volgt per vraag een korte conclusie.

1. Wat zijn data, metadata, model en metamodel?;

Om deze eerste onderzoekvraag te beantwoorden is er eerst gezocht naar bestaande informatie over deze termen. Hiervoor heb ik verschillende bronnen geraadpleegd zoals, wetenschappelijke artikelen, Internet en boeken. De gevonden informatie van bestaande onderzoeken zijn meestal gericht op een specifiek onderwerp zoals databases, data warehousing enz. De simpele omschrijving van data is gegevens, de gegevens die beschikbaar zijn. De data in een database, zijn de gegevens in een database. Zo zijn bijvoorbeeld de naam van een persoon en geboortedatum allemaal data. Metadata zijn de eigenschappen van een bepaald object, zoals titel van een document. Een model is een systeem, een voorbeeld die de werkelijkheid probeert te beschrijven of na te bootsen. Een metamodel is een model van een model, maar op een hoger abstractie niveau.

Voor deze scriptie vond ik het belangrijk om eerst de kennis van andere auteurs als basis te gebruiken. Hierdoor kreeg ik een beter beeld van wat de termen precies betekenen en waarvoor ze gebruikt kunnen worden. De uitwerking hiervan is te vinden in de hoofdstukken 2, 3, 4 en 5, waarbij hoofdstuk 2 een algemene inleiding is voor deze scriptie. Voor een meer gedetailleerde uitwerking en betekenis van de termen, wordt verwezen naar de hoofdstukken 3, 4 en 5.

2. Wat is een mogelijke definitie voor de termen metadata en metamodel?;

Voordat er een definitie gegeven kon worden aan de termen metadata en metamodel is er eerst uitgebreid onderzocht naar de betekenis van data, metadata, model en metamodel. Onderzoeksvraag 1 is hierdoor ook de basis voor de beantwoording van deze tweede onderzoeksvraag.

Om een definitie te kunnen geven van de termen, metadata en metamodel, is er eerst gekeken of de bestaande definities compleet of goed genoeg waren voor deze scriptie. Bestaande definities zijn definities die gegeven zijn door andere auteurs. Deze bestaande definities, zijn dan ook als basis gebruikt voor de beantwoording van deze vraag. Om te bepalen of een definitie goed genoeg of compleet is voor het onderzoek in deze scriptie,

zijn de bestaande definities vergeleken met het resultaat van onderzoeksvraag 1. Geconcludeerd kon worden dat de bestaande definities niet helemaal compleet waren en dus ook niet gebruikt konden worden voor deze scriptie. Dat de bestaande definities niet compleet waren voor deze scriptie betekend niet dat de definities fout zijn, maar dit kwam vooral omdat de auteur zich dan op een specifiek onderwerp heeft gericht en de gegeven definitie aangepast is aan het onderwerp van het onderzoek. Een definitie zoals data over data voor de term metadata en model van een model voor metamodel zijn niet fout, maar eigenlijk helemaal correct. Maar de definitie is zo simpel dat het weer helemaal niks zegt over de term. Met data over data weet een persoon nog steeds niet wat metadata is. Toch kan er begrepen worden waarom de termen metadata en metamodel vaak met deze definities worden omschreven. Metadata en metamodellen zijn niet simpel uit te leggen en er valt heel veel onder de term. Voor dit onderzoek was een specifieke definitie niet goed genoeg en is er aan de hand van de bestaande definities en de opgedane kennis in de literatuur zelf definities gegeven aan de termen metadata en metamodel. De volgende definities voor metadata en metamodellen zijn gehanteerd in deze scriptie:

Definitie metadata:
"Metadata zijn de eigenschappen van een gegeven informatie zowel fysiek als elektronisch"

Definitie metamodel:
"Metamodel is een model gemaakt op een hoger abstractie niveau die de regels en constructies (syntax en semantiek) van het onderliggende model aangeeft"

3. Hoe worden deze termen momenteel toegepast in de praktijk?;

Om metadata en metamodel beter te kunnen begrijpen zijn onderzoeksvragen 1 en 2 nog niet genoeg. Hiervoor zijn ook de bestaande praktische toepassingen van de termen nodig. Om een beeld te kunnen vormen van metadata en metamodellen is er gekeken naar de verschillende toepassingen in de praktijk waarbij metadata en metamodellen gebruikt zijn. In de scriptie zijn maar een beperkt aantal voorbeelden gegeven, maar deze voorbeelden zijn wel gericht op belangrijke onderwerpen die in de praktijk voortdurend aan wijzigingen onderhevig zijn, zoals databases, Internet en bestandssystemen. De gegeven voorbeelden dienen niet alleen voor het uitleggen van de termen, maar zijn ook gebruikt om de waarde van metadata en metamodellen beter weer te kunnen geven. De voorbeelden geven aan hoe belangrijk metadata en metamodellen zijn bij het vinden van nieuwe oplossingen / technologieën om een bepaald probleem op te lossen, zoals bij schema onafhankelijk. Metadata en metamodellen zijn de basis waarop nieuwe ideeën, theorieën en technieken gebaseerd worden.

De praktijkvoorbeelden van metadata zijn te vinden in hoofdstuk 3 en 4. Waarbij hoofdstuk 3 meer gericht is op verschillende toepassingen zoals bestandssystemen, mp3 en fotobestanden. Hoofdstuk 4 is een meer gedetailleerde uitwerking van metadata bij databases. De praktijkvoorbeelden van metamodel zijn te vinden in hoofdstuk 5.

4. Bestaat er een relatie tussen deze termen?;

Onderzoeksvragen 1, 2 en 3 dienden als basis voor de beantwoording van deze onderzoeksvraag. Om te kunnen bepalen of er een relatie bestaat tussen de termen, data, metadata, model en metamodellen, was het eerst nodig om te weten wat ze precies zijn, wat hun betekenis is, wat de definities zijn en hoe ze gebruikt worden in de praktijk. Na de beantwoording en uitwerking van de voorgaande onderzoeksvragen ben ik me gaan verdiepen in de relatie tussen de termen.

Een metamodel is een model van een model met een hoger abstractie niveau. Een model is een voorbeeld van de werkelijkheid en de werkelijkheid bestaat uit data. In het model komen dus ook objecten voor. Metadata zijn de eigenschappen van objecten. Dit geeft al aan dat deze termen allemaal een relatie hebben met elkaar. Het moeilijke was om deze relatie aan te kunnen geven.

De beantwoording van deze vraag is dat er wel degelijk een hiërarchische relatie bestaat tussen de termen, data, metadata, model en metamodel. Het was niet een eenvoudige taak om deze link te leggen, maar het doel was om eerst te kunnen begrijpen wat metadata en metamodellen zijn. De betekenis van de termen vormden een basis waarmee de link tussen de termen vastgelegd kon worden. De relatie tussen de termen vormde zich naarmate ik bezig was met de voorgaande onderzoeksvragen.

5. Wat is de relatie tussen deze termen?;

Na constatering van de relatie tussen de termen, is er ook aangegeven wat de relatie is tussen de termen. De relatie is aan te geven in een hiërarchische structuur waarbij er gebruik wordt gemaakt van het OMG vier lagen model. De relatie tussen de termen is als volgt aan te geven: data is een instantie van een model. Aangezien er bij een model onder anderen de metadata wordt gemodelleerd is data dan ook een instantie hiervan. Model en metadata zijn dan beide een instantie van het metamodel die weer uit meta-metadata bestaat. Met andere woorden een Mn model is een instantie van een Mn+1 laag model. 1) In een n-laag architectuur model M0, M1 … Mn-1, zijn alle elementen van een Mm laag model een instantie van exact één element van een Mm+1 laag model, en voor alle m<n-1 en 2) elke relatie anders dan de “instantie van” relatie tussen twee elementen X en Y suggereert die level(X)=level(Y).

De relatie tussen de termen is uitgewerkt in hoofdstuk 6. Hier zijn de termen bij elkaar gebracht en is dit hoofdstuk dan ook een paraplu waaronder alle termen vallen. Voor de uitgebreide uitwerking van deze onderzoeksvraag wordt verwezen naar hoofdstuk 6. De voorgaande hoofdstukken die een beantwoording of uitwerking zijn van de voorgaande onderzoeksvragen dienden als basis voor de uitwerking van deze onderzoeksvraag.

6. Hoe kan het verband, de relatie tussen deze termen worden weergegeven?;

Bij de voorgaande onderzoeksvragen is er geconstateerd dat er een verband is tussen de termen. Dit verband / relatie is beschreven in onderzoeksvraag 5. De volgende stap was om deze relatie ook vast te kunnen leggen. De paraplu zoals beschreven bij onderzoeksvraag 5, is een model met alle genoemde termen. In deze scriptie is ervoor gekozen om verschillende modellen te maken waarbij de relatie aangegeven wordt. Het ene model is abstracter dan het andere, maar ze kunnen gebruikt worden om de relatie tussen de termen stap voor stap aan te geven.

Er is geprobeerd om vanaf hoofdstuk 5 een begin te maken aan de relatie tussen de verschillende termen. In hoofdstuk 6 is het gehele concept verder uitgewerkt. Het uiteindelijke resultaat is het overkoepelende model dat weergegeven is in Figuur 6-9, Figuur 6-10 en Figuur 6-11.

7.2 Beantwoording van de centrale vraagstelling

De hoofdvraag luidde: "*Hoe kunnen metadata en metamodellen als basis dienen voor het genereren van de gewenste informatie?* "

De data, techniek en de taal zijn de aspecten die een rol spelen bij het genereren van de gewenste informatie, maar het is metadata en metamodellen die het uiteindelijke verband aan kunnen geven. Metadata en metamodellen vormen de link tussen de data, de techniek en de verschillende talen.

De beantwoording van de centrale vraagstelling heeft geresulteerd in een overkoepelend model waar al de termen, data, model, modelleertaal en metamodel in terug komen. Dit overkoepelende model geeft een hiërarchische samenhang weer tussen al deze termen waarbij er ook gebruik is gemaakt van het OMG vier lagen model. Dit model is weergegeven in hoofdstuk 6, Figuur 6-10 Abstract (hiërarchisch) overkoepelend model. Een meer gedetailleerd model waar ook de genoemde voorbeelden in deze scriptie zijn genoemd is te vinden in Figuur 6-11.

Het overkoepelende model kan naar mijn mening als basis gebruikt worden voor alle nieuwe en bestaande technologieën waarbij de techniek moet helpen bij het genereren van de gewenste informatie. Het overkoepelende model is een fundament die door iedereen in de IT wereld gebruikt kan worden. Het overkoepelende model helpt niet alleen bij het begrijpen van de relatie tussen de termen, maar ook bij de betekenis van de termen. Het goed kunnen toepassen van metadata of metamodellen impliceert dat de betekenis van de termen ook duidelijk en begrepen moeten zijn.

Het overkoepelende model is geen oplossing, maar een hulpmiddel voor toekomstige technologieën. Om de gewenste informatie te kunnen genereren is het overkoepelende model niet de directe oplossing, de applicatie moet nog gemaakt en ontwikkeld worden. Gedurende het traject van een project kunnen er nog op verschillende niveaus fouten gemaakt worden waarbij de gewenste informatie niet gegenereerd zal worden. Het niet goed in kaart kunnen brengen van de eisen en wensen, of programeer fouten, ontwerp fouten, modelleer fouten enzovoorts, kunnen allemaal als gevolg hebben dat de applicatie niet zal doen wat gewenst is. Zelfs als er begrepen wordt wat metadata en metamodellen zijn en deze goed toegepast zijn kunnen er nog fouten en problemen voorkomen. Zolang de mens in het project zit, zullen er fouten gemaakt worden, de mens is nou eenmaal niet perfect.

7.3 Suggesties voor verder onderzoek

Gedurende dit onderzoek zijn er een aantal onderwerpen / vragen naar voren gekomen die relevant zijn voor een verder onderzoek. Deze zullen in de volgende subparagrafen worden aangegeven.

7.3.1 WWW

Zoals eerder aangegeven in de scriptie zijn er geen specifieke oplossingen gegeven die gebaseerd zijn op metadata. Zo is de kwestie Internet een steeds groeiend probleem, niet omdat het Internet niet goed is, maar omdat er een overload is aan informatie. Iedereen die iets te vertellen heeft kan dit doen via het Internet. Het wordt elke dag moeilijker om goede en relevante informatie te vinden op het World Wide Web. Metadata schemas zoals MOD, Dublin Core enz, kunnen een oplossing zijn voor nieuwe technologieën die gericht zijn op dit gebied. In deze scriptie heb ik dit probleem maar in het kort kunnen aankaarten. Het is echter een groot probleem die veel meer aandacht verdient, vooral omdat we allemaal gebruik maken van het Internet en we in de toekomst misschien nog afhankelijker ervan zullen zijn.

7.3.2 Metamodel oplossingen (mapping tussen modellen m.b.v. metamodellen)

In paragrafen 5.8 en 5.9 komt het onderwerp van mapping tussen modellen met behulp van metamodellen aan bod. Paragraaf 5.8 gaat over OMG waarbij er gebruik wordt gemaakt van een meta-metamodel (MOF) om modellen te kunnen mappen. Paragraaf 5.9 maakt gebruik van een totaal ander techniek om de mapping tussen de metamodellen en uiteindelijk ook modellen modegelijk te maken. De overeenkomst tussen deze twee theorieën is dat ze allebei ervan uitgaan van een soort hoger gelegen top waar de metamodellen en dus ook modellen met elkaar vergeleken kunnen worden. Bij OMG wordt deze top het meta-metamodel genoemd (MOF) en bij de andere wordt het de root genoemd. Data- en applicatie integratie zijn processen die afschrikwekkend kunnen zijn. Een voorbeeld is gegeven bij schema afhankelijk. Mapping tussen verschillende metamodellen en modellen kunnen een oplossing zijn op dit gebied. In deze scriptie heb ik dit onderwerp opgenoemd als voorbeeld, maar dit onderwerp dient verder onderzocht te worden.

Hoofdstuk 5 behandelt twee verschillende concepten voor het vergelijken en mappen van metamodellen en dus ook modellen. Het gaat hier om: het metamodel architectuur van OMG vs. Het metamodel hiërarchie: een framework voor informatiesystemen. De theorie van OMG wordt in de praktijk al toegepast. Deze twee theorieën kunnen meegenomen worden voor verder onderzoek. Dit is een belangrijk onderwerp en aangezien niet alle modellen in de praktijk MOF conform zijn, kan het metamodel hiërarchie wellicht een oplossing zijn hiervoor.

7.3.3 Federale databases (schema afhankelijk / onafhankelijk)

Mapping tussen modellen zoals aangegeven in voorgaande paragraaf hangt ook nauw samen met federale databases. Het gaat om integratie van verschillende schemas die samengebracht worden tot een metaschema. Metadata en metamodellen spelen hierbij een belangrijke rol en ook dit onderwerp is maar weinig aan bod gekomen in deze scriptie.

Het onderwerp is alleen gebruikt als een voorbeeld, maar in de werkelijkheid is data integratie nog een groot probleem die verder onderzoek verdient.

7.3.4 Algemene suggesties

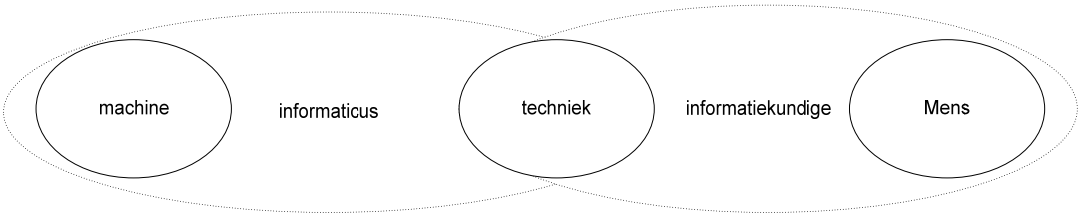
In deze scriptie is het onderwerp nog heel erg theoretisch. Er zijn wel een aantal formele onderbouwingen gegeven, maar deze zijn nog lang niet af. Het werk zoals gepresenteerd in deze scriptie kan voor verder onderzoek nog formeel bewezen worden. Ook praktisch gezien is het onderwerp nog niet bewezen of toegepast.

Er wordt elke dag wel een nieuwe term bedacht voor dingen die misschien net niet vallen onder metadata of metamodel. Bij de afronding van deze scriptie kunnen hierdoor bepaalde aspecten niet meer kloppen of is het overkoepelende model niet compleet meer. Het overkoepelende model is niet praktisch toegepast. Hierdoor kan een verder onderzoek in praktische en formele zin een goed vervolg zijn van deze scriptie.

Verder is MOF niet geheel uitgewerkt in deze scriptie. Er kan hiervan nog voorbeelden gegeven worden. Het idee van MOF is theoretisch behandeld in deze scriptie, maar voor een verder onderzoek kan dit onderwerp praktisch behandeld worden.

7.4 Informaticus / informatiekundige

De informatiekundige wordt vaak de tolk genoemd tussen de organisatie en de programmeur. Een gek idee is dit ook niet. De informatiekundige vertaalt de wensen en eisen van een klant voor de informaticus en omgekeerd vertaalt de informatiekundige de technische aspecten van de informaticus voor de klant. De informatiekundige speelt hierdoor een heel belangrijke rol bij de ontwikkeling en implementatie van applicaties.



Figuur 7-1 relatie informatiekundige en informaticus

Een succesvolle afronding van een (technisch) project hangt voor een groot deel af van een goede samenwerking tussen de informatiekundige, informaticus en de klant. De informatiekundige moet hierdoor de taal van de informaticus kunnen praten en begrijpen. Uiteraard hoeft de informatiekundige niet zelf een applicatie te programmeren, maar moet wel de stof kennen om erover te kunnen praten.

Deze scriptie is minder technisch en is hierdoor een informatiekunde scriptie. Het onderwerp van deze scriptie is zowel voor de informatiekundige als de informaticus relevant. De informatiekundige en de informaticus kijken waarschijnlijk anders naar de onderwerpen metadata en metamodellen. Metadata en metamodellen spelen een steeds groter en belangrijke rol bij het ontwikkelen van nieuwe software applicaties en nieuwe technologieën. Zoals eerder aangegeven was de programmeur vroeger niet

verantwoordelijk voor het genereren van de juiste informatie, als de applicatie maar deed wat ervan verwacht werd. Tegenwoordig weet men beter dat een applicatie niet zomaar informatie genereert en dat het niet om elke informatie gaat, maar de gewenste en dus ook belangrijke informatie die besluiten in een organisatie zal ondersteunen. De programmeur is niet alleen verantwoordelijk voor het genereren van de juiste informatie, maar hij is ook degene die de relevantie van metadata en metamodellen moet zien en toepassen. Hier gaat het dus ook om het begrijpen van de waarde van deze termen en het goed toepassen ervan. De informatiekundige is ook verantwoordelijk voor het generen van de juiste informatie, maar dan in een minder technisch aspect. De informatiekundige zal de applicatie niet ontwerpen en maken, maar zal goed moeten kunnen begrijpen wat informatie is en hoe deze het beste met behulp van metadata en metamodellen gepresenteerd kunnen worden. Het is de informatiekundige die de eisen en wensen van de klant in kaart moet brengen en analyseren hoe informatie getransformeerd in kennis de organisatie zal gaan ondersteunen in haar dagelijkse werkzaamheden.

Het is dus ook voor de informatiekundige belangrijk om te weten wat metadata en metamodellen zijn en wat de mogelijkheden hiervan zijn. En uiteraard zal het verschil tussen de informaticus en informatiekundige zitten in de mate van de technische aard. Een manager zal niet begrijpen hoe ingewikkeld zoiets als data integratie kan zijn, maar de informatiekundige wel en die kan de link zijn tussen de organisatie en de programmeur.

In deze scriptie is er getracht om bij bepaalde onderwerpen een formele onderbouwing te geven. Hoewel de informaticus technischer is en zich veel meer begeeft in het gebied van formalismen, zullen de onderbouwingen in deze scriptie ook voor de informatiekundige begrijpbaar zijn. Er is geprobeerd om door middel van deze formalismen de informatiekundige en informaticus dichter bij elkaar te brengen en op deze manier het onderwerp in deze scriptie voor beide vakgebieden interessant te maken.

7.5 HBO en WO

Twee jaar geleden ben ik begonnen met de opleiding Informatiekunde aan de Radboud Universiteit. Hiervoor had ik mijn vierjarige HBO opleiding Bedrijfskundige Informatica afgerond aan de Hogeschool van Arnhem en Nijmegen. Ook voor mijn afstuderen bij het HBO moest ik een onderzoek verrichten en aan het eind een scriptie schrijven. Er zijn echter grote verschillen in deze scriptie en mijn HBO scriptie. Niet alleen de scripties hebben grote verschillen, maar mijn ervaring op beide scholen is verschillend.

Op het HBO wordt er veel meer gericht op de praktijk en dit zie je ook in de vorm van les geven. We kregen veel meer practica en minder theoretische lessen. Het voordel hiervan is dat je als student wordt voorbereidt op wat er gaat komen en dat is werken in een bedrijf. Na het solliciteren, begint het echte werk en dan moet je ook echt iets doen en niet achter een bureau zitten en alleen maar de theorie doornemen. Op de universiteit heb ik veel meer geleerd hoe je iets aan gaat pakken, bijvoorbeeld een project. Op het HBO heb ik geleerd dat je aan het begin van een project een plan van aanpak maakt, maar het is pas hier op de universiteit dat ik geleerd heb hoe ik om moet gaan met een echt onderzoek. Elk project heeft een probleemstelling, doel en iets dat je wilt verbeteren in de vorm van een nieuwe database, applicatie etc. Maar een plan van aanpak is veel meer dan alleen een planning maken met de standaard tekst die ik geleerd heb op het HBO. Hier op de universiteit heb ik geleerd om op een wetenschappelijke manier na te denken over het

doel, probleem, vraagstelling en aanpak van een onderzoek. Het grootste verschil ligt in het wetenschappelijk omgaan met projecten, onderzoeken en bronnen.

Een ander verschil tussen het HBO en WO, is dat ik met deze scriptie geleerd heb om wetenschappelijke artikelen te gebruiken om mijn onderzoek meer waarde te geven. Voor mijn HBO scriptie heb ik ook een onderzoek verricht, maar het is niet onderbouwt. Het is niet dat ik zelf alles moest kunnen bewijzen, maar als je van andere wetenschappers hun kennis kan gebruiken, dan kan je theorie bewezen worden via eerder gehouden onderzoek die al bewezen is. Dit brengt mij ook op het volgende punt; referenties.

De laatste jaren komt het onderwerp plagiaat vaak in het nieuws voor. De scholen proberen ook op allerlei manieren te controleren of een stuk tekst echt origineel is of gekopieerd is van een ander bron. Er worden zelfs applicaties ontworpen om dit sneller te kunnen controleren met gebruik van het Internet. Maar het probleem komt van twee kanten. Op het HBO heb ik niet geleerd hoe ik met referenties moet werken. Want het kopiëren van een stuk tekst met de nodige referenties in je rapport is legaal en mag dus ook. Een rapport kan meer inhoud en waarde krijgen als je bijvoorbeeld een citaat van een andere wetenschapper gebruikt, maar dan ook zegt dat het van die persoon afkomstig is. Dus waarom zouden we dit niet mogen doen? Maar op het HBO krijg je meer de indruk dat je alles opnieuw moet uitvinden. Want het idee of theorie van een andere wetenschapper gebruiken en verder onderzoeken lijkt niet te mogen. Wel heb ik geconstateerd dat dit niet alleen een HBO probleem is, want van andere WO studenten heb ik ook kunnen vernemen dat ze niet weten hoe ze een tekst van een ander auteur mogen gebruiken in een eigen rapport en dat ze bang zijn dat ze hierdoor gestraft kunnen worden. Dit heeft het gevolg dat een student liever zelf alles opnieuw gaat uitvinden, dan dat ze voor plagiator worden uitgemaakt.

Op de universiteit heb ik ook geleerd dat er zelfs voor de bronvermelding een standaard bestaat. Dat had ik eerder niet gedacht. Maar ook hieraan is er op het HBO geen aandacht besteed.

Deze scriptie heeft om meerdere redenen een WO niveau en geen HBO niveau. Niet alleen om bovengenoemde verschillen, maar ook de formele onderbouwingen in deze scriptie geven mijn werk een WO niveau. Formele onderbouwingen zoals ik gekregen heb bij beweren en bewijzen en formeel denken, heb ik op het HBO niet gekregen. De formele notaties vond ik hierdoor in het begin ook moeilijk en het toepassen ervan vind ik nog steeds moeilijk. Hierdoor zullen sommige formele onderbouwingen misschien nog niet helemaal compleet zijn, maar het is wel een aardig begin.

Op het WO heb ik geleerd om goed na te denken over mijn eigen werk. Wat wil ik precies bereiken, wat is het doel en wat is mijn gewenste resultaat. Bij de gekregen vakken wordt deze manier van denken ook van ons verwacht. Op het HBO ging het meer om het project af te maken, en niet verder nadenken.

7.6 Evaluatie van het traject

Mijn ervaring is dat dit een hele zware, irritante, saaie, leuke, interessante en positieve leerervaring is geweest. Het begin van het traject is het moeilijkst in de zin dat je een probleemstelling moet kunnen aangeven voor je scriptie. Het is vooral zwaar als je niet precies weet wat je wilt. De eerste fase van dit onderzoekstraject was voor mij niet heel

erg moeilijk, maar wel zwaar. Goede en interessante artikelen vinden over metadata en metamodellen is niet altijd even makkelijk. Vervolgens komt het lezen van de artikelen om de interessante stukken voor mijn scriptie eruit te halen. Dit is erg zwaar werk, want van sommige artikelen is maar één alinea belangrijk en die moet je zien te vinden tussen al die pagina's.

De volgende fase is om met de verkregen informatie "iets" mee te doen in de scriptie. Dit is het moment waarop ik creatief moest zijn. Is er verband tussen de termen, hoe vind ik deze en hoe zal ik het aangeven etc. Dit was voor mij de moeilijkste fase. Soms ging het iets makkelijker dan andere dagen. Maar soms zat ik uren lang achter de computer en had ik maar twee zinnen opgeschreven. Het is een kunst om elke dag op te staan en opnieuw door te gaan met de scriptie, terwijl je het gevoel hebt dat je bent vastgelopen. Het is juist op die momenten dat ik geen zin meer had, maar achteraf valt het allemaal weer mee.

Toen ik aan deze scriptie begon, wist ik zelf niet zoveel over metadata en metamodellen. Voor mij was het dus ook helemaal nieuw. Soms werd ik juist helemaal in de war gebracht door informatie die ik gevonden had over deze onderwerpen. Hoe meer informatie ik vond, hoe meer ik in de war raakte. Maar op een gegeven moment ben ik gerichter gaan zoeken voor mijn scriptie en kon ik weer mijn draai vinden.

Soms had ik het gevoel dat er geen einde was voor deze scriptie, maar je moet gewoon doorgaan.

Voor dit onderzoek is er gekozen voor een breed onderwerp waardoor het geheel heel moeilijk was. Bij het kiezen voor een specifiek onderwerp voor een onderzoek, kan de auteur gerichter zoeken. De auteur houdt zich dan alleen bezig met dat onderwerp en verdiept zich in de stof. In dit geval betreft het onderwerp metadata en metamodellen. Het resultaat van dit onderzoek is een overkoepelend model waar alle termen onder vallen. Het moeilijke van het overkoepelende model is, dat je van alles iets moet weten om een geheel plaatje te kunnen vormen. En het is moeilijk om alles dat te maken heeft met metadata en metamodellen te behandelen. Niet alleen moest ik proberen om van metadata allerlei belangrijke en relevante informatie te vinden, maar ook voor metamodellen moest ik dit doen. En metadata en metamodellen is een heel breed onderwerp waar van alles onder valt.

Als Curaçaoënaar is het schrijven van een rapport altijd weer een uitdaging. Juni 1999 ben ik hier in Nederland komen wonen en eerlijk gezegd vind ik de grammatica van de taal het moeilijkst om te beheersen. Deze scriptie is voor mij een leerervaring, maar het kan gebeuren dat er nog grammatica fouten in de scriptie voorkomen of misschien de constructie van een zin niet helaal correct loopt. De beheersing van de Nederlandse taal is een proces waar ik nog steeds mee bezig ben. Maar wat mij altijd heeft geïntrigeerd is dat ook Nederlanders moeite hebben met de taal. Ik moet nog altijd lachen waneer een Nederlander vergelijkt bij een vergrotende trap met 'als' en niet 'dan'. Op zulke momenten weet ik dat ik niet de enige ben die fouten maakt.

Lijst van figuren

Lijst van tabellen

Referenties

[1] Bakker, J., G., M., Data Management is Metadata Management Trendrapportage Data Management Rijkswaterstaat, 3 juni 2002, http://www.wadi.nl/datamanagementismetadatamgt.html

[2] Bentley, John., E., First Union National Park, Metadata: Everyone talks about it, But what is it?, Datawarehousing and Solutions.

[3] Brand, Amy, Daly, Frank en Meyers, Barbara, Metadata Demystified, A guide for publishers, Published by NISO Press, National Information Standards Organisation, 2001, www.niso.org

[4] Brasethvik, Terje, A Semantic Modeling approach to Metadata, Depertment of Computer and Information Science, IDI, Norwegian University of Science and Technology, NTNU, April 1998.

[5] Breton, E., Bézivin, J., Towards an understanding of Model Executability, Ogunquit, Maine, USA, October 2001.

[6] Clark, T., Evans, A., Sammut, P., Willans, J., Applied Metamodelling: A foundation for language Driven Development Version 0.1, www.xantium.com, 21 Dec 2005

[7] Coles, C., Cabinnet office, e-Government metadata, Metadata & search presentations, Contentmanagement for the public sector, 23 november 2004.

[8] Date, C., J., An introduction to Database Systems, sixth edition. SOFSEM'98 Theory and Practice of Informatics: 25th Conference on Current Trends in Theory and Practice of Informatics, Jasna, Slovakia, November 21-27, 1998 Proceedings (Lecture Notes in Computer Science S.), 60-62

[9] Elmasri, R., Navathe, S., B., Fundamentals of Database Systems, Third Edition, Addison-Wesley, December 5th, 2002, 569-583.

[10] Gaarder, J., Sophie's World, Farrar, Straus and Giroux Inc, 1994.

[11] Grabt,J., Litwin, W., Roussopoulos, N., d Sellis, T. 1993. Query languages for relational multidatabases. VLDB J. 2, 153-171.

[12] Halpin, Terry, An ORM Metamodel, Journal of Conceptual Modeling, October 2000, issue 16

[13] Halpin, terry, An ORM metamodel of Barker, Journam of Conceptual Modeling, December2000, issue 17.

[14] Henderson-Sellers, Brian, Atkinson, C., Kühne, T., Gonzalez-Perez, C., Understanding metamodelling, ER 2003 15 october 2003.

[15] Heylighen, F., Complexity and Information Overlaod in society: Why increasing efficiency leads to decreasing control, April 12, 2002, The information society, http://66.249.93.104/search?q=cache:7Ismf4nNZg8J:pespmc1.vub.ac.be/Papers/Info-Overload.pdf+information+overload+%2B+waddington&hl=en&ct=clnk&cd=9&client=firefox-a

[16] Hodge, Gail, Metadata Made Simpler, Published by NISO Press National Information Standards Organisation, 2001, www.niso.org

[17] Hodge, Gail, Understanding metadata Published by NISO Press National Information Standards Organisation, 2001, www.niso.org

[18] Jernst, what are the differences between a vocabulary, a taxonomy, a thesaurus, on ontology and a meta-model?, January 15th, 2003, www.metamodel.com

[19] Lakshmanan, L.V.S., Sadri, F., en Subramanian, S.N. 1997. Logic and algebraic languages for interoperability in multidatabase systems*, The journal of logic programming.

[20] Lakshmanan, L.V.S., Sadri, F., en Subramanian, S.N. 2001. SchemaSQL-An extension to SQL for multidatabase interoperability. ACM trans.Datab.Syst. 26, 4 (Dec), 476-519.

[21] Lans, R., F., van der. 1999. Het SQL leerboek. 65-67, 385-389.

[22] Oei, J., L., H., en, Hemmen, L., J., G., T., van. en, Falkenberg, E., D., en, Brinkkemper, S., The Meta Model Hierarchy: A Framework for Information Systems Concepts and techniques. July 29th, 1992. 2-21.

[23] OMG, Meta Object Facility (MOF) specification, version 1.3, new edition march 2000, 10 mei 2006, www.omg.org.

[24] Pelzer, T., System Tables, 12 juni 2006, http://www.dbazine.com/db2/db2-disarticles/pelzer4

[25] Proper, E., Weide, Th., P., van der, Modelling as Selection of Interpretation, Institute for Computing and Information Sciences, Radboud University of Nijmegen.

[26] Seiner, S., Robert. DBAs Don't need Meta Data. TDAN.com Issue7.0 Articles – December 1998.

[27] Seiner, S., Rober. Meta Data Themes: The Basics. The Data Administrator Newsletter, June 1998.

[28] SIMIN, Begrippenlijst metadata, 17 april 2006, http://www.simin.nl/index.php

[29] Smit, Frans, Het Historisch Data Warehouse, February 2002, http://www.cultivateint.org/issue6/warehouse-d/

[30] Song, H., Metamodels for Object Role Modeling,, march 2005.

[31] Tannenbaum, Adrienne, Metadata Solutions, Using Metamodel, Repositories, XML and Enterprise Portals to Generate Information on Demand, Addison-Wesley, 2nd printing, February 2006.

[32] Waddington, P., Dying for information? A report on the effects of information overload in the UK and WorldWide, Reuters united Kingdom, 1996, http://www.cni.org/regconfs/1997/ukoln-content/repor~13.html#34

[33] Website British Library, Exchange Formats, 17 April 2006, http://www.bl.uk/services/bibliographic/exchange.html

[34] Website Codd's 12 rules, 12 juni 2006, http://en.wikipedia.org/wiki/Codd%27s_12_rules

[35] Website Obituary: Dr. Edgar F. Codd, 12 juni 2006, http://66.249.93.104/search?q=cache:NvMiVDcQ36YJ:www.nao.org.uk/intosai/edp/intoit_articles/18p60top62.pdf+Dr+F+edgar+Codd+%2B+12+rules&hl=en&ct=clnk&cd=2&client=firefox-a

[36] Website Dublin Core, 17 april 2006, http://dublincore.org.

[37] Website, EXIF metadata, http://exif.org/specifications.html

[38] Website, Federated database system, 26 juli 2006, http://en.wikipedia.org/wiki/Federated_database

[39] Website Glossary, Define Integration, 17 april 2006, http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci212359,00.html

[40] Website IBM Federated Database Technology, 13 juni 2006, http://www-128.ibm.com/developerworks/db2/library/techarticle/0203haas/0203haas.html

[41] Website ID3v2, 10 mei 2006, http://www.id3.org/

[42] Website Metadata, 9 maart 2006, http://en.wikipedia.org/wiki/Metadata

[43] Website METS, Metadata Encoding & Transmission Standard, 17 April 2006, http://www.loc.gov/standards/mets/

[44] Website Semantics, www.whatis.com

[45] Website TDAN.com Issue 17.0 Articles, Tannenbaum, Adrienne, A Disaster Crying for Solutions, juli 2001, www.tdan.com

[46] Website, UML Sematics, 4 juli 2006, http://etna.int-evry.fr/COURS/UML/semantics/semant2.html

[47] Website Unmanaged metadata, 3 mei 2006, http://www.e-consultancy.com/newsfeatures/22715/unmanaged-metadata-threatens-quality-of-corporate-decision-making-says-ovum.html

[48] Website, What is metamodeling and what is it good for, http://www.metamodel.com

[49] Website, WinFS overview, http://www.ntfs.com/winfs_basics.htm

[50] Wyss, M., Catharine. and Robertson, L., Edward. 2005. Relational Languages for Metadata Intergration. ACM Transactions on Database Systems, Vol. 30, No. 2, June 2005, 624-600

[51] Zhao, J., L., Schema coordination in federated database management: a comparison with schema integration., Decision Support Systems 29, (1997), 243-257.

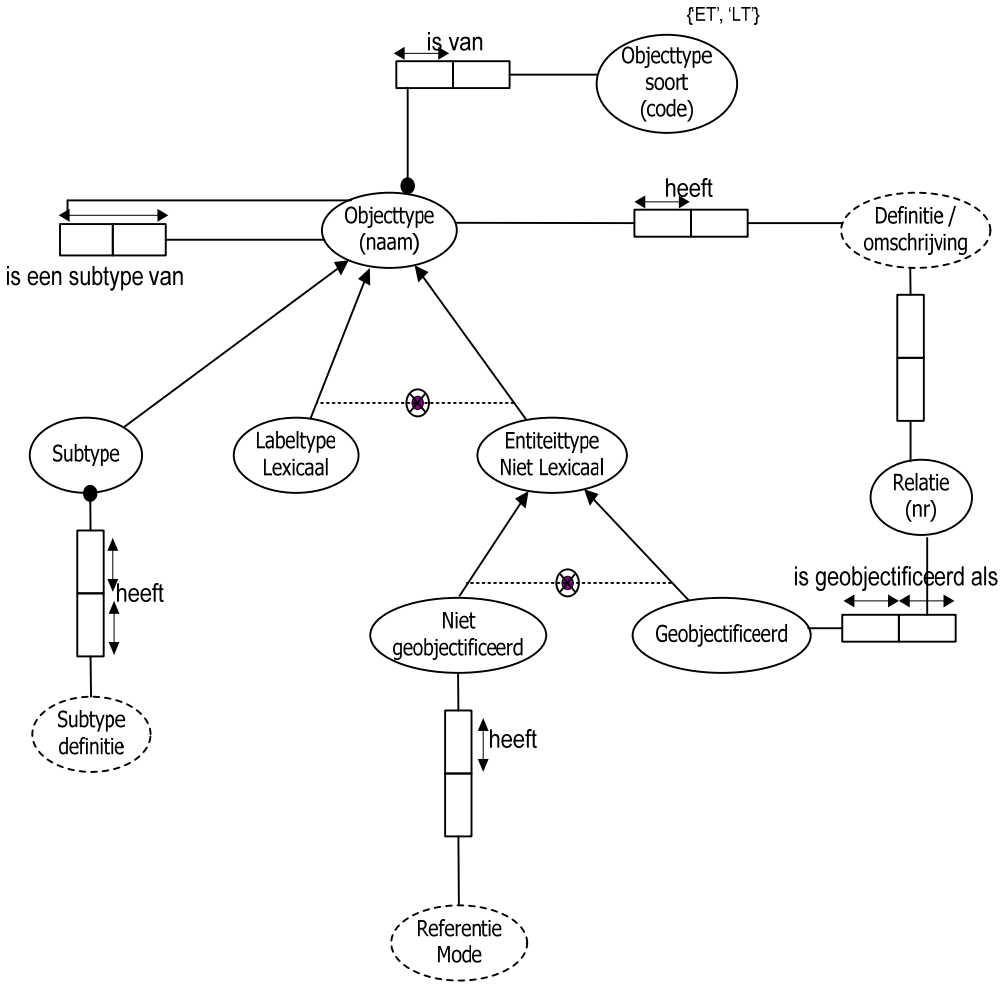
Appendix A: Woordenlijst

Catalog :	Bevat de metagegevens die nodig zijn voor het functioneren van het DBMS
CORBA :	Common Object Request Broker Architecture, een standaard van de OMG voor communicatie tussen gedistribueerde objecten.
Data :	Een gegeven, informatie, gegevens
DBA	Database administrator
DBMS	Database Management Systeem, programma die de opgeslagen gegevens in een database beheert.
DCF	Design rule for Camera File system
DD (Data Dictionary) :	Bevat metagegevens die vooral bedoeld zijn voor ontwerpers, gebruikers en systeembeheer.
EXIF :	Exchangeable Image File Format
FAT	File Allocation Table, is een bestandssysteem ontwikkeld voor MSDOS en WINDOWS.
Federale database :	Een geïntegreerde data repository van verschillende (mogelijk) heterogene data bronnen die gepresenteerd zijn met een consistente en coherente semantiek.
HTML	HyperText Markup Language, is een taal voor de opmaak van web documenten.
ID3 :	Staat toe dat metadata zoals titel, artiest, album en tracknummer opgeslagen wordt in het MP3 bestand
Integratie :	Proces waarbij verschillende componenten samensmelten tot een geheel.
Interoperabiliteit :	Betreft de technische interfaces die het mogelijk maken om onafhankelijk ontworpen producten aan elkaar te knopen.
Mapping :	Een mapping is een set regels en technieken om een model te kunnen omzetten naar een ander model.
Meta schema / scheme :	Zijn sets van metadata elementen, ontworpen voor een specifiek doel, bijvoorbeeld om een specifiek type informatie object te beschrijven.

Metadata :	Metadata zijn de eigenschappen van een gegeven informatie / object zowel fysiek als elektronisch.
Metamodel :	Een metamodel is een precieze definitie van de constructies en regels die nodig zijn voor het creëren van semantiek modellen.
MFD	Material Flow Diagrams
Model :	Een model is een voorbeeld van de werkelijkheid, een systeem dat de werkelijkheid probeert te beschrijven of na te bootsen.
MOF :	Meta Object Facility, een meta-metamodel, gedefinieert door de OMG, dat als basis dient voor zaken als CWM, UML, XMI, CORBA, enz.
NTFS	New Technology File System, is een bestandssysteem gebruikt door Windows NT en opvolgers.
OLTP :	Online Transaction Processing
OMG :	Object Management Group, een internationale onafhankelijke organisatie dat zich tot doel stelt om integratie problemen op te lossen door open, leverancier onafhankelijke specificaties / standaarden te leveren.
Ontologie :	Ontologie beschrijft de eigenschappen van het geheel van dingen waarvan aangenomen wordt dat ze bestaan.
ORM	Object Role Modeling
Portabiliteit :	Portabele programma's kunnen verplaatst worden naar een nieuw systeem, zonder dat er wijzigingen gemaakt moeten worden.
Semantiek :	De semantiek is de wetenschap die zich bezig houdt met de betekenis van constructies.
SQL	Structured Query language, is een ANSI/ISO standaard taal voor een relationele DBMS.
Syntax :	Zijn de regels die de structuur van een taal bepalen. Het geeft aan hoe woorden en symbolen samen worden gevoegd tot een zin of uitspraak.
TIFF :	Tagged Image File Format
UML :	Unified Modeling Language, een object georiënteerde analyse en ontwerptaal, ontwikkelt door OMG.

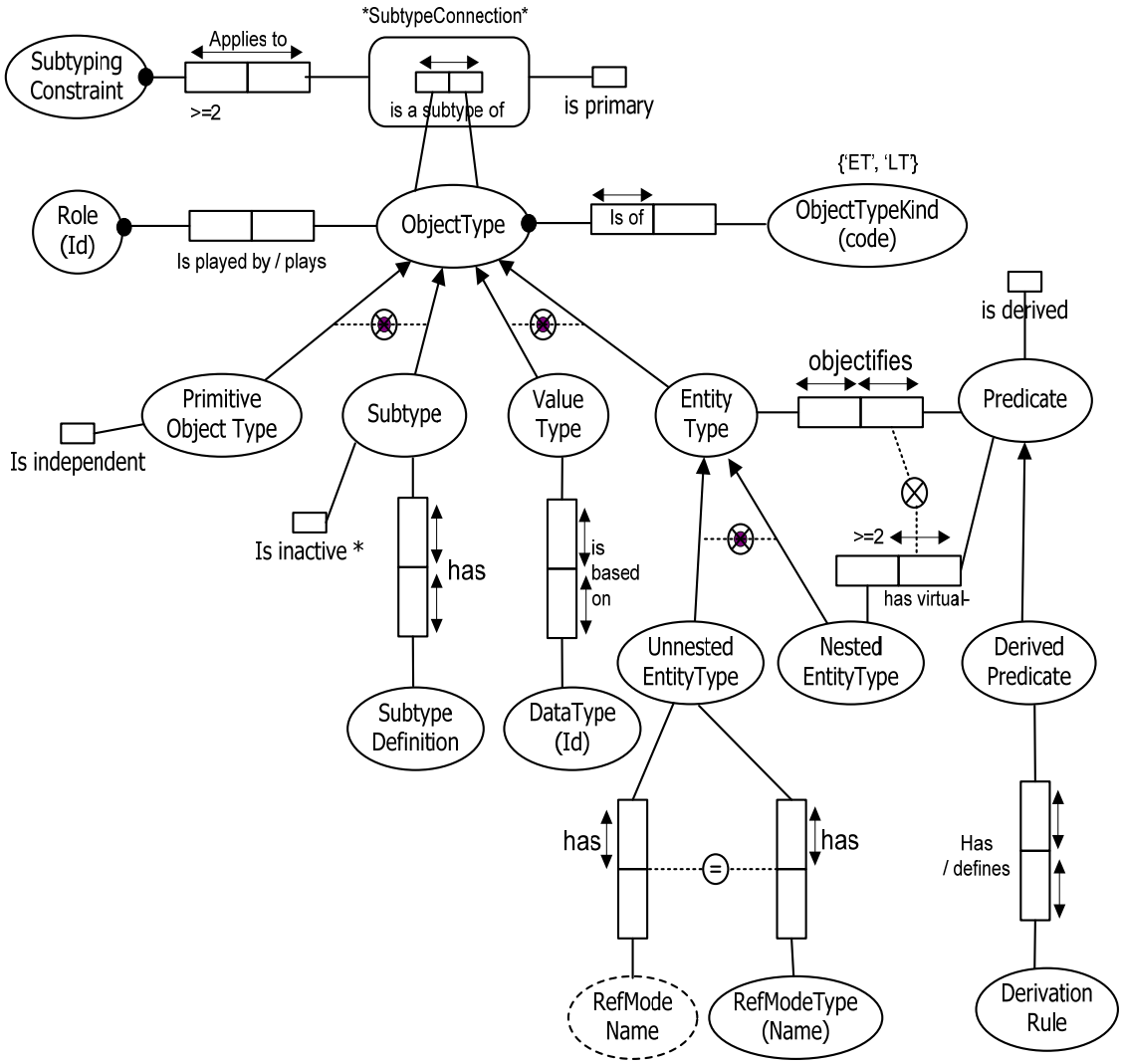
W3C :	World Wide Web Consortium, een organisatie dat zich tot doel stelt technologieën op de markt te brengen die de ontwikkeling van het Web bevorderen. Zij bevordert dus interoperabiliteit.
Wrapper :	Een wrapper is een ontwerp patroon, waarop een code toelaat om verschillende klassen met elkaar te laten werken die normal niet compatibel zijn.
XML :	Extensible Markup Language, door de W3C in 1996 geïntroduceerde taal om data elementen te definiëren in web documenten.
Repository :	Een opslagplaats

Appendix B: metamodel ORM 1



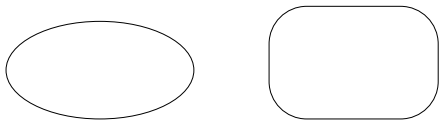
Mogelijk metamodel van ORM, gebaseerd op Haplin, Terry [12]

Appendix C: Metamodel ORM 2

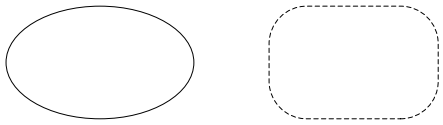


Mogelijk metamodel van ORM, gebaseerd op Song, H., [30]

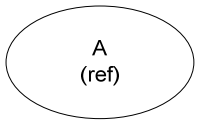
Appendix D: Legenda symbolen ORM



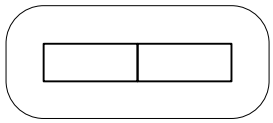
Entiteittype



Labeltype

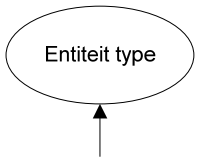


Entiteit met referentie mode



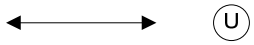
Geobjectificeerde predikaat

Rol



Subtype

Totaliteit constraint

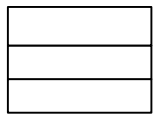


Uniqueness constraint

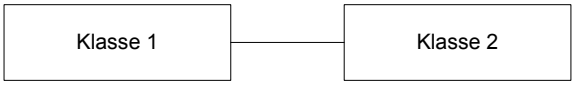
Uitsluiting

Frequency / cardinaliteit constraint

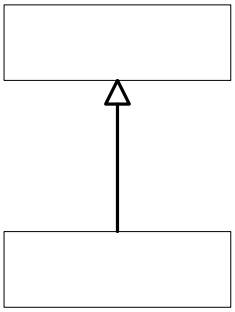
Appendix E: Legenda symbolen UML



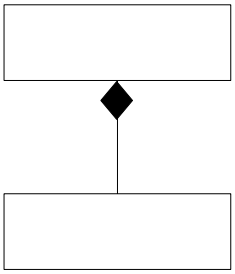
Klasse



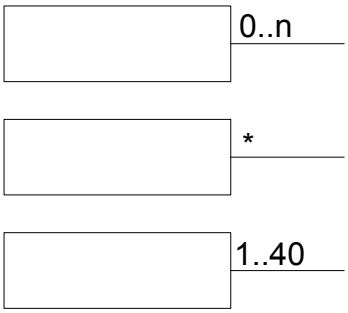
Associatie



Generalisatie



Compositie



Cardinaliteit