ALTERRA
WAGENINGEN UR

IDsW Objectencatalogus

Onderzoek naar de toepasbaarheid van een semantisch datamodel als overkoepelende Aquo standaard

Alterra-rapport 2027
ISSN 1566-7197

E. Verhelst, J.D. Bulens, A. Ligtenberg en S.B. Hoek

IDsW Objectencatalogus

Dit onderzoek is uitgevoerd In opdracht van IDsW in het kader van het onderzoek naar gebruik van semantisch model coderingen voor de IDsW Aquo Standaarden.

Projectcode 5237360

IDsW Objectencatalogus

Onderzoek naar de toepasbaarheid van een semantisch datamodel als overkoepelende Aquo standaard

E. Verhelst, J.D. Bulens, A. Ligtenberg en S.B. Hoek

Alterra-rapport 2027

Alterra Wageningen UR
Wageningen, 2010

Referaat

E. Verhelst, J.D. Bulens, A. Ligtenberg en S.B. Hoek, 2010. *IDsW Objectencatalogus; Onderzoek naar de toepasbaarheid van een semantisch datamodel als overkoepelende Aquo standaard*. Wageningen, Alterra, Alterra-rapport 2027. 62 blz.; 22 fig.; 5 tab.; 7 ref.

Op internet worden semantische technieken gebruikt om data op een betekenisvolle manier te ontsluiten. Deze technieken zijn in dit onderzoek gebruikt om vast te stellen of zij een bijdrage kunnen leveren aan de standaardisering van gegevens die worden gebruikt bij watermanagement.

Trefwoorden: thesaurus, ontologie, semantiek, datamodellering, standaarden, waterbeheer, gegevenshuishouding.

ISSN 1566-7197

Dit rapport is gratis te downloaden van www.alterra.wur.nl (ga naar 'Alterra-rapporten'). Alterra Wageningen UR verstrekt geen gedrukte exemplaren van rapporten. Gedrukte exemplaren zijn verkrijgbaar via een externe leverancier. Kijk hiervoor op www.boomblad.nl/rapportenservice.

© 2009 Alterra
Postbus 47; 6700 AA Wageningen; Nederland

© 2009 Alterra
Postbus 47; 6700 AA Wageningen; Nederland
Tel.: (0317) 480700; fax: (0317) 419000; e-mail: info.alterra@wur.nl
Mede auteurs: M. Danes, J. Franke, I. La Riviere

Naamsvermelding-Niet-commercieel-Gelijk delen 3.0 Nederland
Meer informatie over deze Creative Commons licentie op:
http://creativecommons.org/licenses/by-nc-sa/3.0/nl/

Alterra aanvaardt geen aansprakelijkheid voor eventuele schade voortvloeiend uit het gebruik van de resultaten van dit onderzoek of de toepassing van de adviezen.

Alterra-rapport 2027
Wageningen, mei 2010

Inhoud

Samenvatting

IDsW beheert en ontwikkelt informatiestandaarden voor het Nederlandse Waterbeheer. Deze standaarden worden gebruikt voor verschillende specifieke doelen. Elke standaard heeft daardoor een eigen formaat. Er is behoefte aan een betere technische afstemming tussen deze standaarden, waardoor het beheer en het gebruik van de standaarden wordt vereenvoudigd.
In dit onderzoek is uitgezocht of technieken die gebruikt worden bij de ontsluiting van informatie op internet gebruikt kunnen worden om dit vraagstuk op te lossen. Het gaat hierbij om semantische uitwisselingsstandaarden.
In dit onderzoek is aangetoond dat deze technieken een wezenlijke bijdrage kunnen leveren aan de totstandkoming van de verbetering van de koppeling tussen de bestaande standaarden. Met name de definitie van en de relatie tussen in de standaarden gebruikte begrippen kunnen eenduidiger worden vastgelegd. Er is meer onderzoek nodig om vast te stellen in hoeverre deze technieken in de praktijk kunnen worden gebruikt. Wij denken in het bijzonder dan aan het waarborgen van de integriteit van de gegevens.

1 Inleiding

IDsW beheert en ontwikkelt informatiestandaarden voor het Nederlandse Waterbeheer. Deze standaarden hebben de overkoepelende naam 'Aquo standaard'. Binnen de Aquo standaard bestaan standaarden voor specifieke doelen (uitwisseling, opslag en definitie). De informatie binnen de standaarden is op elkaar afgestemd, maar er zijn verschillen in structuur en gebruikte technieken voor vastlegging. Dit heeft als gevolg dat het beheer van de standaarden complex en tijdrovend is. Hierdoor bestaat de behoefte aan een complete en eenduidige overkoepelende begrippendefinitie en -structuur ofwel een abstracte basisstandaard (hierna te noemen 'objectencatalogus').

IDsW biedt de standaarden aan via een open interface (XML, XSD), zodat gebruikers deze zonder gegevensverlies in toepassingen kunnen gebruiken. De objectencatalogus moet aansluiten bij deze werkwijze.
De gegevens die in de objectencatalogus zullen worden opgeslagen zijn primair bedoeld om data in de zin van 'begrippen' vast te leggen en te koppelen. Technieken die worden gebruikt bij de totstandkoming van het semantisch web zijn hier specifiek voor ontwikkeld.

IDsW heeft Alterra de opdracht gegeven te onderzoeken in hoeverre semantische gegevensmodellen kunnen worden gebruikt voor de ontwikkeling van een objectencatalogus. Dit rapport bevat de verslaglegging van dit onderzoek.

De onderzoeksopdracht is projectmatig uitgevoerd met als projectresultaat naast dit document een ontwerp van de objectencatalogus op basis van semantische datamodellen en een website en een webservice die het model ontsluiten. Dit document bevat tevens de verslaglegging van de bouw van de objectencatalogus en de website en webservice.

1.1 Leeswijzer

Dit document kan het beste als volgt worden gelezen. Hoofdstuk 2 schetst de context van het onderzoek, dat wil zeggen het doel, de methode en het resultaat. Hoofdstuk 3 geeft een theoretische achtergrond. Hoofdstuk 4 bevat de feitelijke verslaglegging van het onderzoek en in hoofdstuk 5 staan de conclusies en ons advies.

2 Onderzoek context

2.1 Aanleiding

De verplichting voor het beschikbaar stellen van standaarden brengt met zich mee dat aansluiting gezocht moet worden bij enerzijds gebruikerswensen en anderzijds de nieuwste ontwikkelingen in informatievoorziening. Eén van de opdrachten van IDsW is het integreren van de CIW gegevensstandaard (Commissie Integraal Waterbeheer), het Adventus stelsel, IMWA (Informatiemodel Water) en Omega Woordenboek tot een enkele, integrale standaard. Aan de zijde van de gegevensdefinitie is dit traject inmiddels nagenoeg afgerond. Wat betreft beschikbaarstelling en het gebruiksgemak wil men nog verbeteringen aanbrengen. Dit onderzoek is onderdeel van dit proces.

2.2 Probleemstelling

De Aquo standaarden worden gebruikt voor meerdere toepassingen. Aquo-lex is de bron voor de beschrijving van de definitie van de gegevens. De Aquo domeintabellen worden voor hetzelfde doel ingezet en zijn specifiek bedoeld als (opzoek)lijsten. LM Aquo (Logisch Model) is de bron voor de opslag van de gegevens en IMWA en UM Aquo zijn bedoeld als uitwisselmodel. Het nadeel van van deze verschillende modellen is dat ze elk met een ander doel vor ogen en in een andere tijsdperioden zijn ontstaan. Dit leidt eenvoudig tot de introductie van inconsistenties voor objecten en hun eigenschappen die in deze verschillende modellen voorkomen. Dit komt de bruikbaarheid niet ten goede. Het vinden van de verbanden tussen domeinwaarden en definities is lastig. Deze relaties zijn momenteel niet of nauwelijks voor de gebruiker terug te vinden. De ontwikkeling van een nieuw overkoepelend model dat de relaties tussen de standaarden vastlegt kan hier verandering in brengen. Bij verkenningen naar te gebruiken methoden hiervoor kwam aan het licht dat bij de ontwikkeling van het semantisch web technieken worden gebruikt die mogelijk geschikt hiervoor zijn. Deze technieken concentreren zich op het ontsluiten van relaties en begrippen in een semantische context onafhankelijk van de toepassing of de gebruikte modelleringstechnieken. De bruikbaarheid van deze semantische technieken zijn onderzocht met betrekking tot de wens om relaties en begrippen (semantiek) voor de Aquo standaarden eenduidig en beheersbaar vast te leggen.

2.3 Doel

Het doel van het onderzoeksproject is:

- te onderzoeken in hoeverre het mogelijk is met semantische modellen definities, eigenschappen en onderlinge relaties tussen de concepten uit de huidige Aquo standaarden vast te leggen;
- te onderzoeken op welke wijze de gegevens in een semantische omgeving (de objectencatalogus) kunnen worden beheerd;
- te onderzoeken op welke wijze de gegevens in de objectencatalogus kunnen worden gekoppeld aan de bestaande Aquo standaarden.

2.4 Onderzoeksvragen

De centrale vraag van het onderzoek is in hoeverre semantische modellen geschikt zijn voor de implementatie van het concept van de objectencatalogus. Teneinde deze vraag te kunnen beantwoorden zijn de volgende deelvragen geformuleerd:

- Zijn er bestaande en gepubliceerde formele ontologieën die een voorbeeld kunnen zijn voor de objectencatalogus en/of waarvan een deel kan worden gebruikt voor de objectencatalogus?
- Welke semantische frameworks, markup languages en modelleringsmethoden zijn geschikt voor het ontwerp van de objectencatalogus?
- Wat is het semantisch model van de objectencatalogus (gebaseerd op het ontwerp op p. 18 van [1])?
  - Kunnen de eigenschappen, constraints en relaties zoals gedefinieerd in het ontwerp op p. 18 van [1] worden toegepast in het semantische model van de objectencatalogus?
  - In hoeverre kan de data van de huidige Aquo standaarden worden geladen in het semantische model van de objectencatalogus?
  - Op welke wijze moet de data in het semantische model van de objectencatalogus worden beheerd?
  - Als de objectencatalogus als invoer bronbestand wordt gebruikt, in hoeverre kunnen dan koppelingen worden gemaakt tussen bestaande Aquo standaarden en de objectencatalogus?
  - In hoeverre kan in het semantische model van de objectencatalogus de historie van de wijzigingen worden vastgelegd?
  - Welke tools zijn beschikbaar voor het beheer van de gegevens in de objectencatalogus?
  - Welke tools zijn beschikbaar voor het beheer van het ontwerp van de objectencatalogus?

2.5 Werkwijze

Het onderzoek is uitgevoerd door middel van een literatuurstudie, aangevuld met een toetsing van de onderzoeksvragen aan de praktijk. Het voorstelontwerp van de objectencatalogus zoals getekend in p. 18 van [1] is uitgewerkt tot een ontwerp van een open semantisch model van de objectencatalogus. De bruikbaarheid van twee ontologie designtools is hierbij tegen het licht gehouden.
De mogelijkheden en beperkingen van het koppelen van de huidige standaarden aan de objectencatalogus is onderzocht via een literatuurstudie en een korte evaluatie van te gebruiken tools. Voor zover mogelijk binnen het tijdsframe van de opdracht is de import uitgevoerd. Over de technische en praktische problemen die zijn gesignaleerd wordt in dit document gerapporteerd.

2.6 Resultaat

Het onderzoek wordt uitgevoerd door het toetsen van de theorie aan de praktijk. Hiervoor is een proof of concept van de objectencatalogus gebouwd waarin werkelijke data van de Aquo modellen is geladen.
Dit model is gebruikt als basis voor de ontsluiting via een website en een webservice.
Van de onderzoeksresultaten is verslaglegging gedaan in dit document.

3 Theorie en concepten

3.1 De begrippen

Voordat we naar de objectencatalogus gaan is het belangrijk enkele begrippen nader toe te lichten die relevant zijn voor het achterliggende concept. In de huidige praktijk heerst veel verwarring rond deze begrippen en worden ze vaak door elkaar gebruikt. De begrippen zijn semantiek, ontologie, taxonomie, lexicografie en thesaurus. De beschrijving van deze begrippen die nu volgt is grotendeels ontleend aan definities en beschrijvingen zoals die in de Wikipedia te vinden zijn. Voor de volledigheid geven we in deze paragraaf meer begrippen weer dan voor dit onderzoek gebruikt zijn.

De semantiek of betekenisleer houdt zich bezig met de betekenis van woorden zoals we deze gebruiken in onze taal. In de informatietheorie wordt onder semantiek de betekenis van geordende opeenvolgingen van informatie verstaan. Formele semantiek is de overkoepelende term voor de manier waarop zowel de semantiek en de logica als de gewone taal en de formele taal (bijvoorbeeld computertaal) wordt beschreven. Aan de basis van al deze systemen ligt hetzelfde, namelijk het gebruik van bepaalde reeksen symbolen (bijvoorbeeld een alfabet) waar door middel van interpretatie een betekenis aan wordt toegekend.

In de informatica en de logica is een ontologie het product van een poging een uitputtend en strikt conceptueel schema te formuleren over een bepaald domein. Een ontologie is typisch een datastructuur die alle relevante entiteiten en hun onderlinge relaties en regels binnen dat domein bevat, zoals bij een domeinontologie het geval is. Het gebruik van het woord binnen de informatica is afgeleid van het van het woord ontologie binnen de filosofie.

Een ontologie verschilt van een databank doordat een ontologie niet alleen feiten bevat maar ook regels, gevat in logische formules. Uit dergelijke regels kan men nieuwe feiten afleiden met een automatisch redeneerprogramma.

Een ontologie die niet verbonden is aan een specifiek domein, maar dat algemene entiteiten probeert te beschrijven, wordt een foundation ontology, top level ontology of upper ontology genoemd. Over het algemeen dienen gespecialiseerdere en domeinspecifieke schema's te worden gecreëerd om de data geschikt te maken voor beslissingen in de werkelijkheid.

In de informatica is een conceptueel schema, of hogere-orde datamodel of conceptueel datamodel een afbeelding van concepten en hun relaties. Bijvoorbeeld een conceptueel schema voor een familie bevat abstracties als vader/moeder, oma/opa, ouder en kind. Een conceptueel schema wordt gebruikt in een eerste fase van een database ontwerp.

Taxonomie is de wetenschap van het indelen. Taxonomie (taxon = groep) verwijst naar zowel de classificatie van dingen als naar de methode die aan de basis van deze classificatie ligt. Vrijwel alles kan taxonomisch worden ingedeeld: levende wezens, planten-gemeenschappen, dingen, plaatsen, gebeurtenissen, enzovoort.

Een thesaurus is een soort naslagwerk. De term komt uit het Grieks θησαυρός (thèsauros) en werd in het Latijn overgenomen als thesaurus (meervoud thesauri) en betekent schatkamer, het weggelegde. Het werd aanvankelijk in de taalkunde opgesteld als een logisch-systematisch (en ook alfabetisch, maar niet verklarend) woordenboek: de begrippen van een taal werden gecategoriseerd en vergeleken met verwante begrippen:

- synoniemen;
- woorden die een ruimer begrip beschrijven: hyperoniemen;
- woorden die een engere betekenis hebben: hyponiemen;
- woorden met tegengestelde betekenis: antoniemen; of
- begrippen die aan het lemma verwant zijn, maar een andere nuance uitdrukken, of een overlappende betekenis hebben.

Een thesaurus gebruikt men om het exacte woord voor een voorwerp (een bepaalde vakterm) of een woord met de gewenste connotatie (uit stijloverwegingen) te vinden.

Uit een Nederlandstalige thesaurus kan men besluiten dat synoniemenlijst een nauwere betekenis heeft dan thesaurus en het begrip lexicon overlapt. Soms hanteert men betekeniswoordenboek als synoniem van thesaurus, maar de betekenis van een woord is in dit bijzondere geval niet gelijk aan de verklaring ervan, maar vloeit voort uit de situering van dit woord binnen het geheel.

De aanduiding 'thesaurus' wordt nu ook gebruikt voor een naslagwerk met geselecteerde woorden of concepten, bijvoorbeeld een gespecialiseerd vocabularium binnen een bepaald interesse- of vakgebied, zoals geneeskunde of muziek.

Met behulp van een thesaurus kan men bijvoorbeeld de catalogus van een bibliotheek beter toegankelijk maken dan door middel van een ordening, die uiteindelijk willekeurig is. Zo is men niet meer strikt gebonden aan de terminologie - en de taal! - van een boek of andere informatiedrager. Per publicatie of informatie-item kan men bij vele thesauri zelfs meerdere descriptoren (thesaurustermen) toekennen. Een itemsgewijze beschrijving wordt zodoende versterkt door een systematische ontsluiting. Er bestaan thesauri voor vele wetenschappelijke disciplines. Voor sommige gebieden zijn er zelfs concurrerende thesauri, bijvoorbeeld voor de kunsten de Art and Architecture Thesaurus en Iconclass. Taxonomieën zijn aan thesauri verwant.

De benaming thesaurus wordt ook gebruikt in de taalkunde voor de grote gegevensbestanden met de woordenschat van het Latijn, de Thesaurus Linguae Latinae te München, en die van het Grieks, de Thesaurus Linguae Graecae van de Universiteit van Californië. Het is dan de 'schatkamer' van oorspronkelijke teksten, waarop men zich baseert om de dode taal (Latijn of Grieks) te reconstrueren.

Thesauri treft men in papieren vorm (boek) aan, maar ook als elektronisch medium.

De term lexicografie (letterlijk: ληξικος γραφειν woord beschrijven) heeft betrekking op het implementeren van lexicologische principes bij het samenstellen van woordenboeken, encyclopedieën, concordanties, etc.

Het draait in de lexicografie in wezen om de vraag: hoe kan woordinformatie (zoals betekenis, spelling, uitspraak en gebruik) op een gebruiksvriendelijke en verantwoorde manier voor de gebruiker toegankelijk gemaakt worden?

De objectencatalogus is lexicografisch en is er een ontologie opgesteld. Een thesaurus of ook wel een taxonomie die de vaktermen beschrijft vormt een onderdeel van deze ontologie.

Voor dit onderzoek hebben we de focus gelegd op het ontwerpen van een thesaurus gebaseerd op een ontologie. De kern is dat daarmee de semantiek wordt vastgelegd. Verder beschrijft deze ontologie nog de context waarin de vaktermen voor het domein Water worden gebruikt en geeft het relaties aan. Op deze wijze kan de objectencatalogus gebruikt worden, zoals dat nu in de huidige praktijk ook gebeurt met LM Aquo, Aquo-lex en UM Aquo. Ook domeintabellen zoals in gebruik bij deze modellen worden in de objectencatalogus beschreven als er in deze tabellen semantische (lexicografische) relaties zijn.

3.2 De samenhang

Bij een objectencatalogus gaat het om de semantiek. Semantiek kunnen we zoals we eerder hebben gezien vastleggen in ontologieën. Het bijbehorende werkveld is wat we in het Engels noemen Knowledge Organizations System (KOS) ofwel het vastleggen van kennis of ook wel het 'organiseren' van kennis in systemen. Belangrijkst kenmerk hierbij is het bepalen van de structuur waarin de kennis betekenis krijgt voor de gebruiker. In het werkveld is er weer een onderverdeling in een aantal typen systemen dat hierbij gebruikt wordt. Als eerste begrippenlijsten (term lists) waaronder woordenlijsten woordenboeken. Verder classificaties of categorieën waaronder onderwerpenlijsten (heading lists), classificatieschema's en taxonomieën en de relatielijsten ('relationship lists'), waaronder thesauri, semantische netwerken en ontologieën vallen. Deze opsomming is nog niet uitputtend, er zijn nog tal van andere mogelijkheden om kennis gestructureerd vast te leggen en te gebruiken. Voor de objectencatalogus voor het domein water zijn vooral de indelingen genoemd bij thesauri en ontologieën naar onze mening van belang. Thesauri en ontologieën zijn nauw aan elkaar verwant, waarbij een ontologie verdergaat dan wat in een thesaurus is vastgelegd.

Zoals al eerder beschreven bestaan er ontologieën voor de verschillende domeinen. Als een ontologie niet specifiek voor een domein geldt zoals de eerder genoemde foundation of upper ontologie dan worden alleen de algemene concepten beschreven die geldend zijn voor alle domeinen. De belangrijkste functie van een dergelijke upper ontologie is dat in de breedte interoperabiliteit wordt ondersteund tussen ontologieën.

Naast een upper ontologie kennen we ook de lower ontologie of ook wel domeinontologie. Domeinontologieën op hun beurt kunnen weer in meerdere soorten worden onderverdeeld. We kennen de taakontologie, waarbij specifieke taken in een proces uitgevoerd worden vastgelegd, een applicatie-ontologie, waarbij voor een specifieke toepassing de vastlegging plaatsvindt en we kennen de echte domeinontologie waarbij de meer statische werkelijke wereld wordt beschreven. Bij de objecten die bepalend zijn voor het waterbeheer gaat het met name om deze laatste soort. Als het gaat om de beschrijving van de objecten dan zou een thesaurus kunnen volstaan, maar in dit onderzoek van de IDsW gaat dit verder omdat ook het gebruik van belang is. Voorbeelden van bestaande thesauri zijn GEMET voor het domein Milieu en AGROVOC voor het Agrarische domein. In de volgende paragraaf gaan we eerst in op de vraag 'waarom een ontologie?'.

3.2.1 Waarom een ontologie?

In de afgelopen jaren is door de ontwikkeling van ontologieën het gebruik steeds meer gemeengoed geworden in de werkomgeving van domeinexperts. Een ontologie als expliciete formele specificatie van termen in een domein met hun relaties wordt steeds meer gebruikt op het Internet. De ontologieën op het Web variëren van een grote taxonomie voor websitecategorieën zoals bijvoorbeeld Yahoo gebruikt, tot lijsten van producten met hun eigenschappen voor de verkoop. Dit wordt bijvoorbeeld gebruikt bij de website van Amazon. Het WWW consortium heeft het RDF (Resource Description Framework) als standaard ontwikkeld om kennis te coderen op webpagina's en het leesbaar te maken voor elektronische ‘agents’ die informatie op het Web doorzoeken. Het Defense Advanced Research Projects Agency (DARPA) in de VS heeft in samenwerking met het W3C OWL (Ontology Web Language) ontwikkeld door RDF uit te breiden met meer mogelijkheden voor expressieve

constructies [8]. In veel disciplines worden nu standaard ontologieën ontwikkeld door domeinexperts om de informatie uit een vakgebied te delen en te annoteren. Een ontologie definieert nu een gemeenschappelijke woordenlijst voor vakgenoten die de informatie in hun domein willen delen. De gekozen structuur maakt het mogelijk voor machines definities van basisbegrippen (vaktermen) te interpreteren met hun relaties in het domein.

Redenen om een ontologie te ontwikkelen zijn de volgende:
1. het delen en begrijpen van de structuur van informatie door mensen en machines;
2. om hergebruik van domeinkennis mogelijk te maken;
3. om aannames binnen een domein expliciet te maken;
4. om domeinkennis te scheiden van operationele kennis;
5. voor uitvoeren van analyses met domeinkennis.

1. Het delen en begrijpen van de structuur van informatie door mensen en machines. Dit is het meest gebruikelijke doel voor het ontwikkelen van een ontologie. Bij bijvoorbeeld verschillende websites in het domein Water die ieder dezelfde services gebruiken voor telkens een ander doel. Als deze websites de termen die gebruikt worden uit een gezamenlijke onderliggende ontologie haalt die gedeeld wordt dan kan software deze begrijpen. Daardoor is daaruit op maat informatie af te leiden die de verschillende vragen van de eindgebruikers kan beantwoorden. Ook wordt deze informatie als invoer voor andere programma's gebruikt.

2. Om hergebruik van domeinkennis mogelijk te maken. Dit is een drijvende kracht achter het ontologie-onderzoek. Als voorbeeld noemen we dat meerdere modellen uit verschillende domeinen het begrip tijd willen representeren. Dit omvat onder meer de notie van tijdsintervallen, tijdstippen, het meten van tijd, etc. Als dit in detail in een ontologie in een domein is vastgelegd dan kan dit door gebruikers uit een andere domein worden hergebruikt. Het achterliggende concept van tijd is meerdere domeinen namelijk gelijk. Op eenzelfde wijze kunnen ook delen van ontologieën uit andere domeinen worden hergebruikt als deze binnen het eigen domein van toepassing zijn.

3. Het expliciet maken van aannames binnen een domein. De vastgelegde concepten vormen de basis voor onderliggende implementaties. Hard gecodeerde aannames over de echte wereld in programma's zijn moeilijk te vinden en te begrijpen maar het is ook moeilijk deze te veranderen. Vooral voor iemand zonder de benodigde programmeerervaring. Aanvullend zijn expliciete specificaties van domeinkennis zeer bruikbaar voor nieuwe gebruikers voor wie het belangrijk is de termen in dat domein te leren en toe te passen.

4. Het scheiden van domeinkennis en operationele kennis is een ander algemeen kenmerk van gebruik van ontologieën. We kunnen beschrijven wat voor de samenstelling van een product de componenten zouden moeten zijn volgend uit de specificaties. Een geautomatiseerd programma kan op basis daarvan een product uit de componenten samenstellen. Het algoritme gebruikt deze ontologie om producten samen te stellen en kan op heel verschillende terreien worden ingezet bijvoorbeeld voor de samenstelling van een PC. Maar we kunnen ook hetzelfde algoritme gebruiken voor een heel ander product, bijvoorbeeld een lift.

5. Uitvoeren van analyse met domeinkennis wordt mogelijk gemaakt juist als een beschrijvende specificatie van de vaktermen beschikbaar is in een ontologie. Vervolgens is een formele analyse van vaktermen ook zeer waardevol als een bestaande ontologie wordt hergebruikt of wordt uitgebreid.

Een ontologie is niet een doel op zich. Het ontwikkelen van een ontologie is toegespitst op de definitie van een set van gegevens met hun structuur (conceoten) voor gebruik in meerdere programma's. Probleemoplossende methoden, domeinafhankelijke applicaties, en software programma's (meer specifiek 'agents') gebruiken

ontologieën en kennisbanken gebouwd op ontologieën voor gegevens. Hierdoor wordt het mogelijk om niet alleen binnen domeinen, maar ook over domeien heen toepassingen te ontwikkeling de de opgeslagen kennis combineert en integreert. Zoals bijvoorbeeld een ontologie voor wijn en spijs het mogelijk maakt om met geschikte combinaties van wijn en gerechten passende maaltijden samen te stellen. Deze ontologie kan dan gebruikt worden als basis voor sommige toepassingen in een reeks 'managementtools' voor restaurants. Een applicatie kan ook wijnsuggesties maken voor het menu van de dag of vragen beantwoorden van obers en klanten. Een andere applicatie kan een analyse maken van de wijnvoorraad in de kelder en aanbevelingen doen welke wijn vervolgens ingekocht dient te worden.

3.3 Het semantisch web

Het semantisch web is volgens de W3C: '… a common framework that allows data to be shared and reused across application, enterprise, and community boundaries' [2]. Het specifieke kenmerk van het semantisch web is dat het een web is van open data en niet van (door een applicatie) gepresenteerde data. Hierdoor ontstaat de mogelijkheid voor derden om de geserveerde databronnen als data te gebruiken in eigen applicaties en te koppelen aan andere open databronnen.

De W3C Semantic Web Activity heeft een aantal specificaties gepubliceerd waarin technieken staan beschreven die de open ontsluiting van databronnen regelen. De belangrijkste is RDF. In dit framework is vastgelegd hoe data kan worden weergegeven op het web. De specificatie van RDF Schema (RDFS) legt de syntax vast van objecten (klassen) en hun relaties. De basisgedachte achter RDF is de triple relatie Subject, Predicate, Object. Deze drie-eenheid is de bouwsteen van alle datamodellen in het semantisch web. Elk onderdeel van een triple (object, predicate en subject) heeft een URI (Uniform Resource Identifier) in de vorm van een http-link. Dit is de unieke sleutel waarmee het kan worden geïdentificeerd en benaderd. Een instantie van een object of subject heet in RDF een individual.

RDF is een simpele manier om gegevens op het web te beschrijven. Niet voor alle toepassingen is dit geschikt. Wanneer zaken als cardinaliteit, classificatie en constraints belangrijk zijn, is de RDF syntax niet toereikend. Voor dit doel is OWL (Ontology Web Language) ontwikkeld. OWL is gebaseerd op RDF en werkt dus ook met triples.

Specifiek voor de ontsluiting van kennis systemen (Knowledge Organization Systems of 'KOS') is SKOS (Simple Knowledge Organization System) ontwikkeld. Typische voorbeelden van een SKOS toepassing is een thesaurus, taxonomie of classificatie schema [5].

Omdat OWL en SKOS gebaseerd zijn op RDF kunnen ze als uitbreidingen binnen een RDF framework worden opgenomen. Op die manier kan een combinatie framework van RDF, OWL en SKOS worden gemaakt. De aparte OWL en SKOS syntax zijn aan hun eigen namespace binnen het framework te herkennen.

Omdat SKOS en OWL eigenschappen hebben die van toepassing zijn op de doelstellingen van de objectencatalogus is onderzocht in hoeverre deze vocabulaires gebruikt kunnen worden in het ontwerp van de objectencatalogus.

Een belangrijk kenmerk van RDF waardoor het zich van andere modelleertalen onderscheid is de relatie oriëntatie (in tegenstelling tot klasse oriëntatie). In RDF kan een predicate (=relatie, property) bestaan zonder dat er een klasse bij hoort. Klassen worden juist gekoppeld door predicates (properties).

Er bestaan verschillende soorten properties. De belangrijkste zijn:
– Datatype Property;
– Object Property;
– Annotation Property.

Een datatype property wordt gebruikt om instanties aan data values (vergelijkbaar met de attribuut waarde in ERD diagrammen) te koppelen. Een object property wordt gebruikt om klassen te koppelen. Dit laatste is vergelijkbaar met het leggen van relaties in een UML diagram. Het soort relatie (zoals in UML bijvoorbeeld aggregatie, compositie) wordt voor een deel bepaald door de constraints die in de object property (object relatie) is vastgelegd (bijvoorbeeld cardinaliteit) maar verder door de naamgeving (semantiek) van de relatie. Zo kan de relatie 'isDeelVan' (in UML de compositierelatie) alleen bestaan als een modelleur hem heeft gemaakt en omdat de gebruiker via de naamgeving weet dat hier de compositierelatie wordt bedoeld.
Er zijn weinig object relaties in de semantische standaarden zoals RDF en OWL gedefinieerd. De belangrijkste zijn de 'is-a' (Engels, 'is een') als definitie van subklasse en 'inverseOf' als inverse relatie.

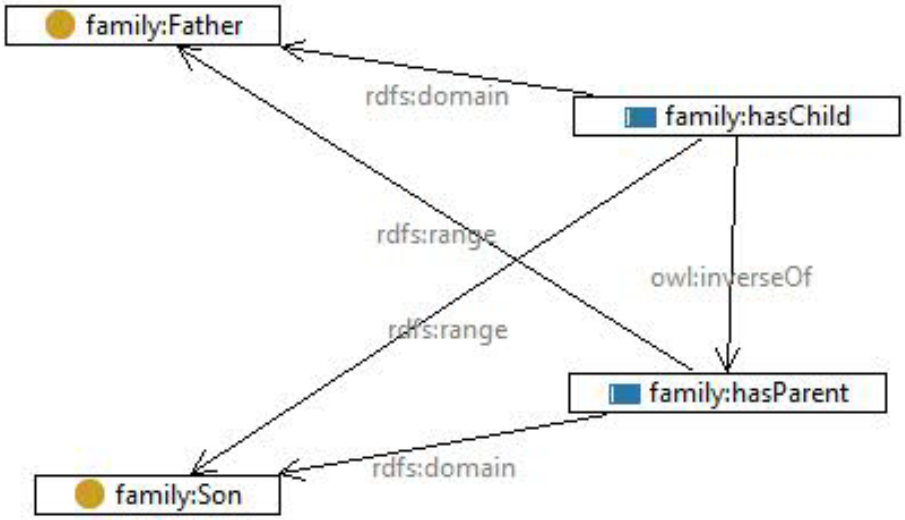
Het bereik van een property wordt door het domain en de range van de property bepaald. Voor een object property is dat: domain klasse - object property - range klasse (bijvoorbeeld Kind heeftOuder Ouder). Voor een datatype property is de range gelijk aan de waarde die de datatype property heeft (string, integer, float, boolean, etcetera). Het domein van een datatype property is de klasse waarvoor het datatype geldt (bijvoorbeeld: 'naam' heeft als domein 'Kind' en als range 'string').

Een inverse property is een property die een omgekeerde domain-range relatie heeft met een andere property. Een voorbeeld is: 'heeftOuder' en 'isOuderVan'. Door ook de inverse relatie van een property vast te leggen wordt een regel in het model opgeslagen die gebruikt kan worden om nieuwe informatie af te leiden. Bijvoorbeeld: stel dat in de data is vastgelegd dat het kind een individual is met een heeftOuder relatie, dan kan uit deze informatie worden afgeleid dat er ook een individual is met een isOuderVan relatie.

Een annotation property is bedoeld om informatief commentaar te verwerken. Het verschil tussen een annotation property en een string datatype property is onder meer dat een annotation property niet gebruikt mag worden in een constraintregel.

Een subklasse beschrijft de 'is-a' relatie. Hieruit volgt dat als iets een subklasse is, dat het dan vanzelfsprekend ook tot de bovenliggende klasse behoort. De properties van de superklasse worden overgeërfd door de subklasse.

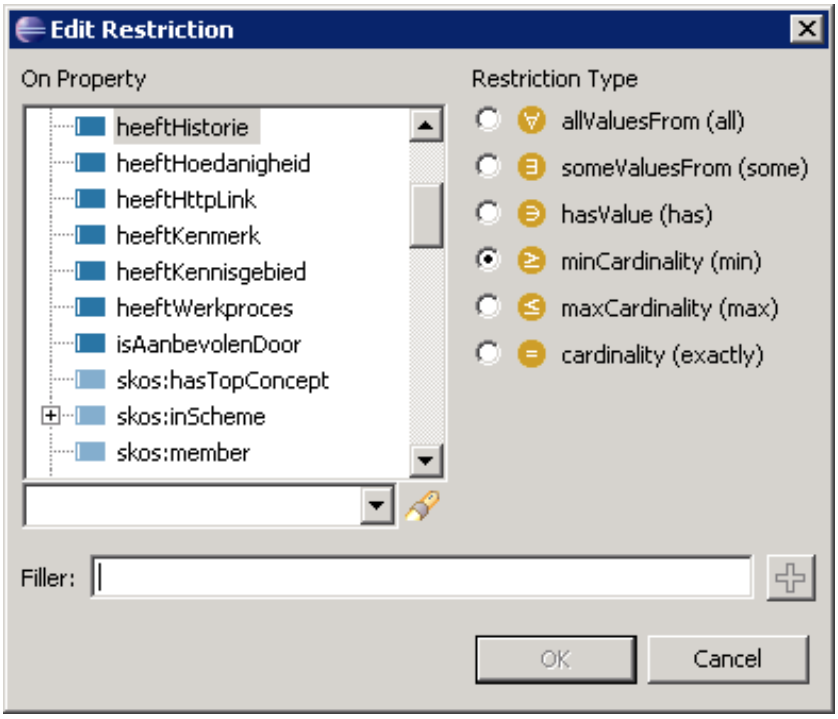
Met een ontologie wordt bedoeld een RDF/OWL bestand dat een groep van begrippen beschrijft inclusief daarin geldende kennisregels. Als voorbeeld wordt vaak de familie ontologie gebruikt [3] (figuur 1).



Figuur 1

Een deel van de family ontologie (de gele objecten zijn klassen, de blauwe zijn properties).

Een OWL ontologie kan cardinaliteit en constraints bevatten. Deze regels worden in het ontwerp opgenomen als restrictie (figuur 2).



Figuur 2

Het toekennen van restricties.

Ook andere restricties zoals bijvoorbeeld de union tussen klassen (AND) en de intersectie (OR) kunnen worden toegekend. Niet alle restricties worden bij invullen gecontroleerd. Bij een zogenaamde consistency check wordt gekeken of de logische constraints in de ontologie op elkaar zijn afgestemd. Voorbeeld: stel dat is gespecificeerd dat een AquoDomeinEenheid een eenheid is van enkel en alleen een Kenmerk dan kan een AquoDomeinEenheid geen eenheid zijn van een Concept: (zie figuur 5)

EntiteitKlasse heeftEenheid AquoDomeinEenheid is geldig
Concept heeftEenheid AquoDomeinEenheid is ongeldig

Een consistency check gebeurt na het invullen van de data door het runnen van een reasoner (zie volgende paragraaf).

3.3.1 Reasoning en Queryen

Een RDF/OWL bestand bevat data die net als bij een database bevraagd kan worden door het uitvoeren van een query. Dit werkt analoog aan query technieken bij een database, alleen de query engine en de query syntax zijn anders. De query taal voor RDF/OWL is SPARQL. Deze taal is vergelijkbaar met SQL voor relationele databases. Een SPARQL query is bijvoorbeeld:

```
SELECT ?x ?y
WHERE {
   ?x skos:prefLabel ?y  .
FILTER (?y = "Vispassage")
}
```

Het resultaat van deze query is de individual waarvan de property skos:prefLabel de waarde 'Vispassage' heeft.

Reasoning is een actie die op een RDF/OWL bestand met data wordt uitgevoerd. Niet alle constraints in een RDF/OWL bestand worden bij data invoer door een ontologie editor gevalideerd. Dit moet ter controle achteraf gebeuren door het aanroepen van een reasoner. In sommige ontologie editors is een reasoner ingebouwd (Topbraid Composer) bij andere editors moet hij extern worden gestart en vanuit de editor worden aangeroepen (vroegere Protégé versies).

Reasoning is zeker niet hetzelfde als queryen. Reasoning is het evalueren van de gevolgen van in de RDF/OWL opgeslagen regels en constraints. Het resultaat van reasoning kan zijn:
a. consistency checking;
b. concept satisfiability;
c. een classificatie;
d. realisatie (inference).

Ad a.
Een reasoner kan de consistency van een RDF/OWL bestand checken. Hierbij wordt de ingevulde data op de constraints getest en bij onjuiste invoer wordt dit gemeld.

Ad b.
Het checken van concept satisfiability wil zeggen dat de reasoner test of een klasse wel individuals kan hebben.

Ad c.
Een reasoner kan ook individuals classificeren. Hierbij worden lege klassen gemaakt die voorzien zijn van een logische omschrijving. Als voorbeeld nemen we de familie ontologie [3]: voor de lege klasse 'Senior' is de volgende logische omschrijving gedefinieerd:

Person hasAge GreaterThan65

De reasoner zal alle individuals die voldoen aan deze regels plaatsen (classificeren) in de lege klasse 'Senior'.

Ad d.
Realisatie of inference wil zeggen dat een reasoner uit regels die zijn opgeslagen in de ontologie een conclusie kan afleiden. Het bekendste voorbeeld is:

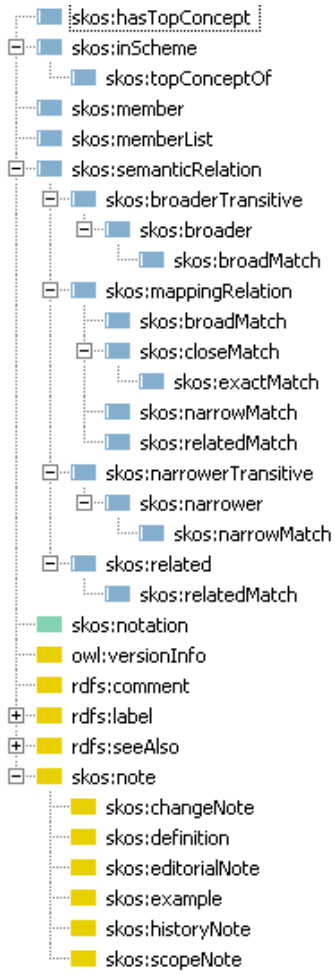
Alle mensen zijn sterfelijk
Socrates is een mens
———
Dus Socrates is sterfelijk

4 Onderzoeksresultaat

In dit hoofdstuk worden de resultaten van het onderzoek gepresenteerd. In paragraaf 4.1 wordt uitgelegd waarom is gekozen voor SKOS en OWL. In paragraaf 4.2 wordt het model van de objectencatalogus besproken en de ontwerpkeuzes worden hier gemotiveerd. In paragraaf 4.3 worden voorbeelden gegeven van hoe de objectencatalogus in de praktijk zal werken en paragraaf 4.4 behandelt de onderzoeksresultaten van het laden van de data in de bestaande standaarden naar de objectencatalogus. In paragraaf 4.5 wordt het beheer van de objectencatalogus behandeld. Ten slotte worden onze ervaringen met het ontsluiten van een OWL file op het internet besproken in paragraaf 4.6.

4.1 De gebruikte datamodellen SKOS en OWL

SKOS is zeer geschikt om beschrijvingen van en verbanden tussen begrippen weer te geven. Wat betreft beschrijvingen zijn er mogelijkheden om naamgeving, definitie, voorbeeld, notes en taal op te slaan. Wat betreft relaties zijn naast hogere orde/lagere orde ook exacte relatie en het algemene 'gerelateerd' aanwezig. Door het beschrijven van de verschillende manieren waarop begrippen gerelateerd zijn kan een semantische laag aan de objectencatalogus worden toegevoegd, die gebruikers in staat stelt meer contextinformatie rond het begrip te verzamelen.
Het SKOS datamodel heeft de volgende properties (figuur 3):



Figuur 3

De SKOS properties.

Deze properties kunnen in principe allemaal worden gebruikt. In de objectencatalogus zijn ze echter niet allemaal gebruikt maar wel de volgende:

Tabel 1

De gebruikte SKOS properties in de objectencatalogus.

SKOS property	Attribuut volgens [1]
skos:prefLabel	Naamgeving + taal
skos:altLabel	Afkorting
skos:editorialNote	Toelichting
skos:definition	Definitie
skos:broader	Hogere orde
skos:narrower	Lagere orde

De relatie 'sterk gerelateerd' en 'zwak gerelateerd', die nodig zijn in de objectencatalogus, zijn niet in SKOS gedefinieerd en daarom apart in de objectencatalogus opgenomen.

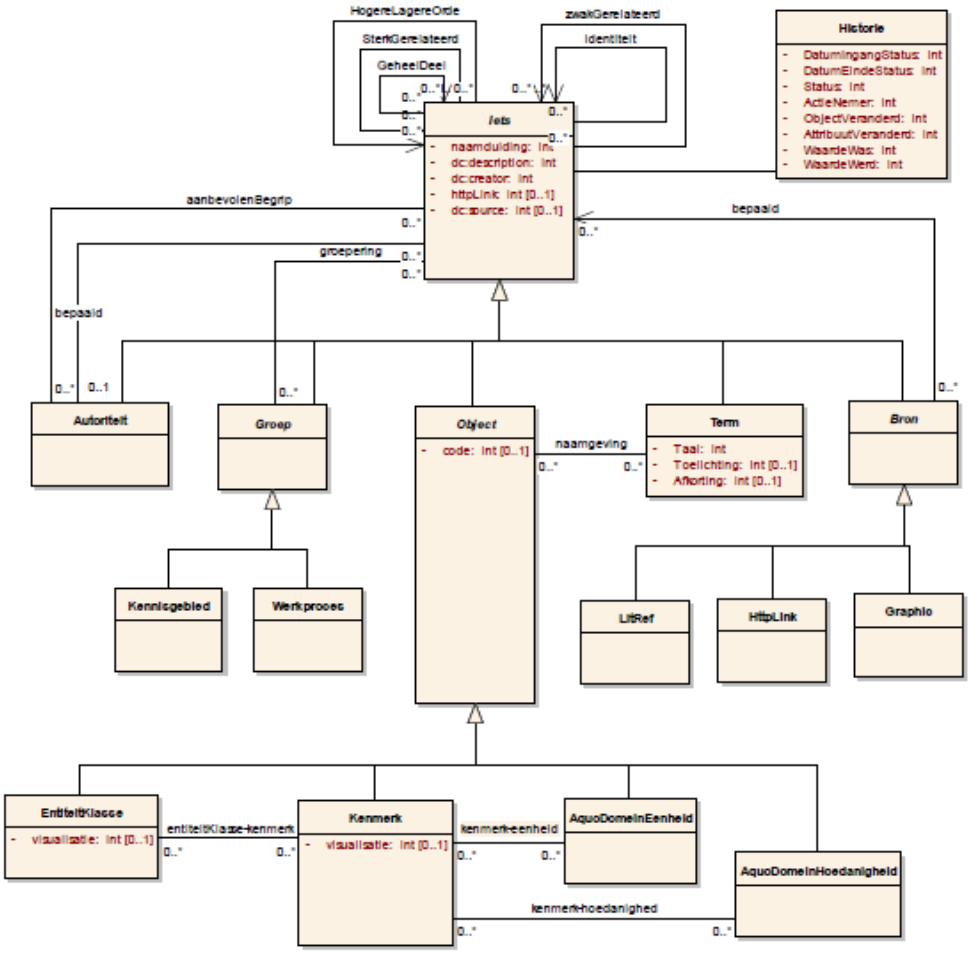
In SKOS moet de taal worden opgenomen. Door meerdere waarden bij prefLabel te specificeren kunnen de vaktermen in de objectencatalogus ook meertalig worden. Dit wordt als volgt genoteerd (**vetgedrukte** tekst):

```
<skos:Concept rdf:ID="CON0006">
   <skos:prefLabel xml:lang="en">wave frequency</skos:prefLabel>
   <skos:prefLabel xml:lang="nl">Golffrequentie</skos:prefLabel>
   <dc:source rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Aquo-lex</dc:source>
   <skos:definition rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >de reciproke waarde van de golfperiode</skos:definition>
   <dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >voor demonstratie objectencatalogus</dc:description>
   <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >CON0006</rdfs:label>
  </skos:Concept>
```

Aangezien SKOS is bedoeld voor toepassingen van kennissystemen zoals Aquo-lex is dit gekozen voor het ontwerp van de objectencatalogus. Waar het SKOS datamodel onvoldoende mogelijkheden bood zijn aanvullingen in gedaan in OWL. Een voorbeeld daarvan is 'sterk gerelateerd' en 'zwak gerelateerd'. Ook zijn de relaties met de standaarden IMWA/UM Aquo en LM Aquo zo specifiek dat daarvoor eigen properties zijn gedefinieerd in een eigen namespace 'catalogus'. Dit wordt beschreven in de volgende paragrafen.

4.2 Het model van de objectencatalogus

Het model is gebaseerd op onderstaand UML model dat afkomstig is uit [1].



Figuur 4

De objectencatalogus volgens [1].

Voor conversie van dit UML model naar SKOS/OWL zijn problemen geïdentificeerd waardoor ontwerpkeuzes zijn gemaakt. Deze problemen kunnen worden onderverdeeld in twee typen:

1. de conversie van UML naar RDF/OWL in het algemeen;
2. het gebruik van SKOS in het bijzonder.

Ad 1.

Zoals eerder geschetst worden in RDF/OWL de relaties anders gedefinieerd dan in UML. Waar in UML de relaties in de modelleertaal worden vastgelegd wordt in RDF/OWL de relatie voornamelijk in de naamgeving vastgelegd (predicate/property). Dit heeft geleid tot de keuze om de associatie tussen de UML klassen te vertalen naar de RDF/OWL relatie 'heeft' gevolgd door de naam van de te koppelen klasse (voorbeeld: Kenmerk heeftEenheid AquoDomeinEenheid).

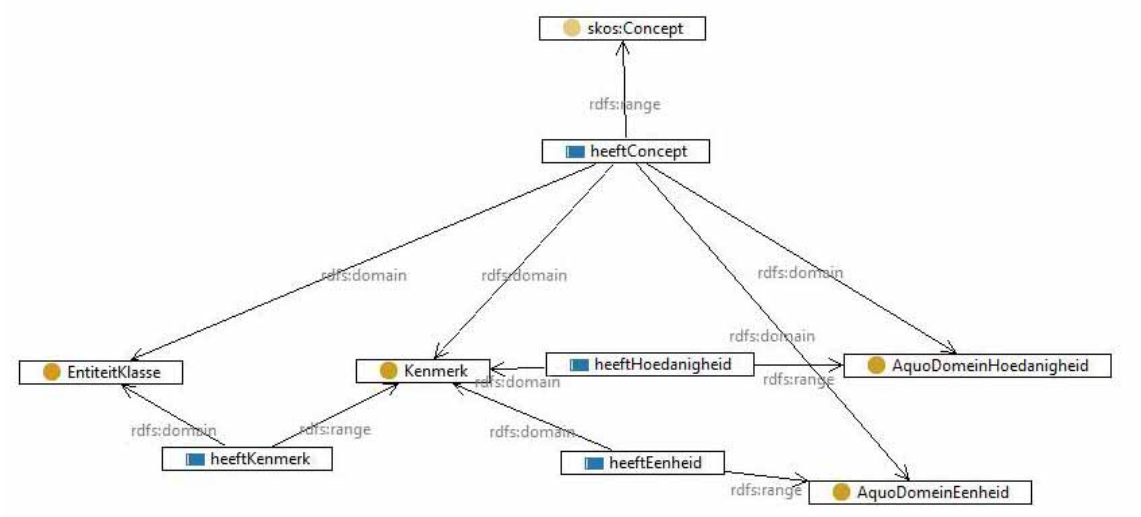
De noodzaak van een abstracte klasse in UML is niet aanwezig in RDF/OWL. In een semantisch model kunnen data en object properties meerdere domein- en rangebereiken hebben. De abstracte klasse 'Object' hoeft daardoor niet in het semantisch model te worden opgenomen.

De klasse Bron is opgenomen met de subklassen LitRef, HttpLink, Graphic en AquoBron omdat hier in principe individuals van Bron kunnen bestaan die een of meerdere waarden in de subklassen hebben (er kan bijvoorbeeld een Graphic, HttpLink of LitRef zijn van een enkele bron). Wat betreft de klasse AquoBron: de oorspronkelijke herkomst van de EntiteitKlassen, Kenmerken, Domeinen en Concepten wordt vastgelegd in de klasse AquoBron. De individuals van AquoBron zijn bijvoorbeeld LM Aquo, IMWA, UM Aquo, Aquo-lex, etcetera.

De klasse Groep is als klasse opgenomen met de gedachte de mogelijkheid open te laten om kennisgebieden en werkprocessen te rubriceren. In het UML diagram is het een abstracte klasse.

De individuals krijgen een unieke sleutel door drie letters en een cijferreeks te combineren. De lengte van de cijferreeks is standaard 4, behalve bij 'Historie' waar de hoeveelheid te verwachten instanties groter is.

Ad 2.
In het model [1] speelt de abstracte klasse 'Iets' een centrale rol. In het door Alterra ontworpen model voor de objectencatalogus is het centrale 'iets' vastgelegd volgens het SKOS concept schema. De SKOS klasse Concept is in dit model de vastlegging van een begrip uit Aquo-lex. De klasse Term wordt dan redundant en verdwijnt hierdoor uit het model.
Alle klassen die onder Object vallen zijn door middel van de property heeftConcept gekoppeld aan de Concept klasse. Dat betekent dat van de EntiteitKlassen, Kenmerken, en domeintabellen Concepten kunnen bestaan. Op deze manier is de koppeling gemaakt tussen Aquo-lex (Concept klasse) enerzijds en LM Aquo (Entiteiten in de EntiteitKlasse klasse), IMWA/UM Aquo (Klassen in de EntiteitKlasse klasse) en de domeintabellen Eenheid en Hoedanigheid (AquoDomeinEenheid en AquoDomeinHoedanigheid) anderzijds. De domeintabellen zijn via de klassen AquoDomeinEenheid en AquoDomeinHoedanigheid met de respectievelijke properties heeftEenheid en heeftHoedanigheid gekoppeld aan Kenmerk. In figuur 5 zijn deze relaties weergegeven.



Figuur 5
De relaties van een concept uit Aquo-lex met gegevens uit de domeintabellen en LM Aquo en UM Aquo.

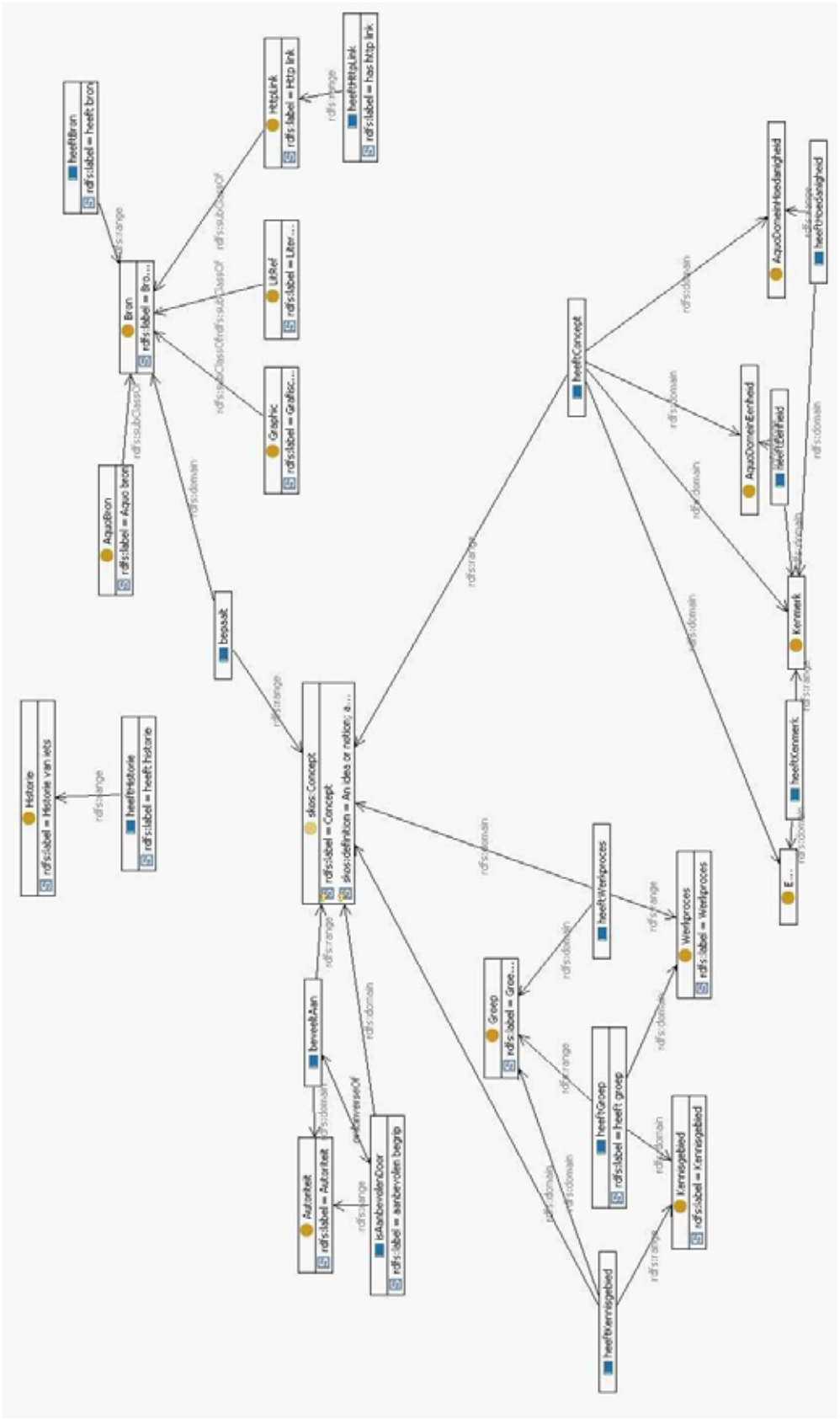
In Aquo-lex worden van een concept ook andere relaties vastgelegd, zoals de bron, het bijbehorende werkproces en kennisgebied en de autoriteit die het concept aanbeveelt. Van deze relaties is een aparte afbeelding gegeven in figuur 6.

Van alle relaties kunnen in principe de inverse relaties ook worden gemodelleerd. Bijvoorbeeld: de inverse relatie van 'heeftConcept' zou zijn 'isConceptVan'. Er is ervoor gekozen om in dit stadium dit niet te doen, omwille van de begrijpbaarheid van het model.

Vanwege de gewenste functie van de objectencatalogus (het vinden van relaties en het beschrijven van vaktermen) is ervoor gekozen om 'Concept' het centrale punt te laten zijn in het semantische model. Dat betekent dat alles vanuit Concept is gemodelleerd en dat er voor alle te koppelen entiteiten uit LM Aquo en klassen uit IMWA/UM Aquo een Concept aanwezig moet zijn. Dit betekent tevens dat er alleen 'broader' en 'narrower' relaties bestaan voor een Concept, en niet voor bijvoorbeeld een EntiteitKlasse.

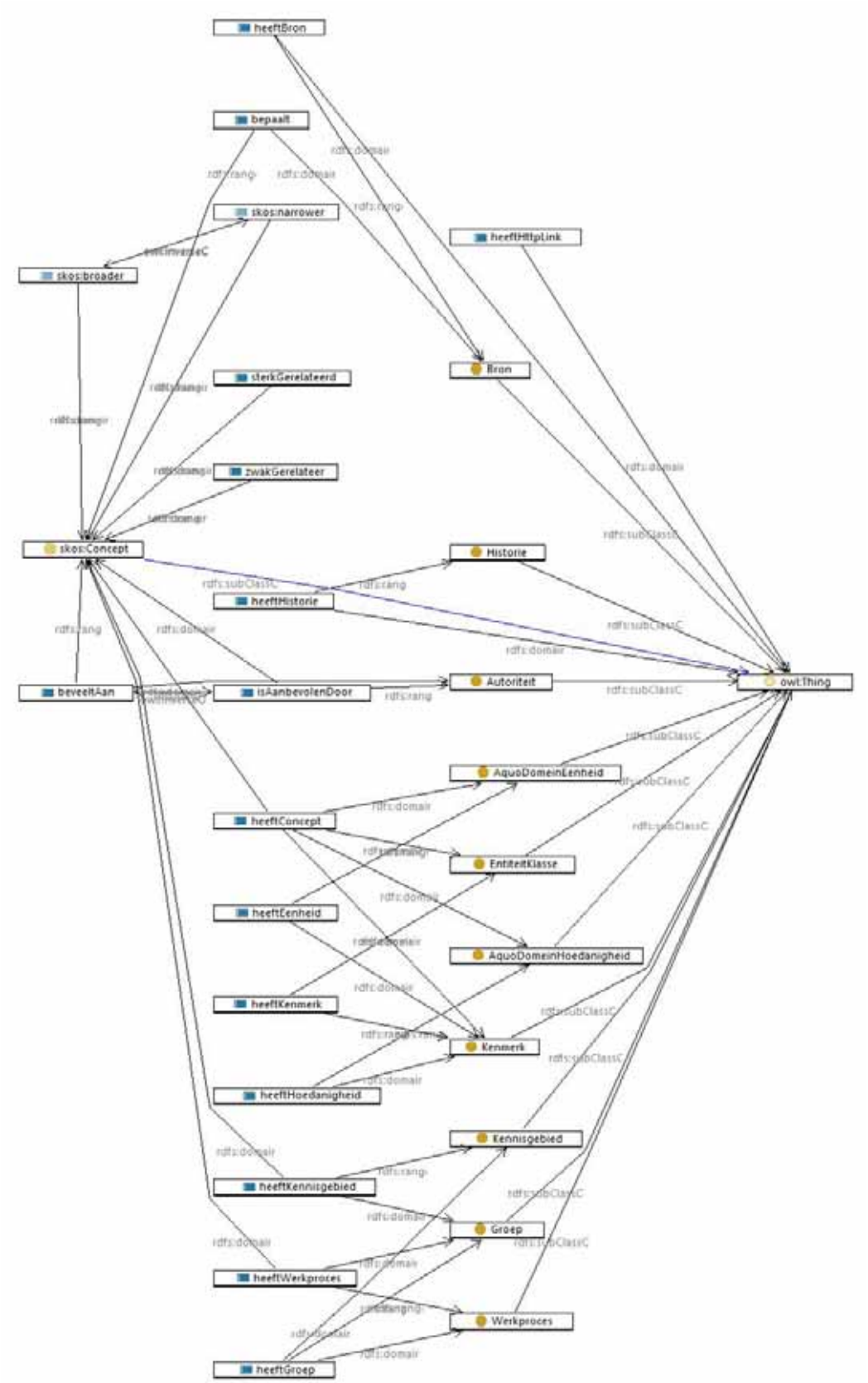
Het SKOS schema bevat naast de Concept klassen nog de klasse ConceptScheme, Collection en OrderedCollection. Deze worden in de SKOS standaard gebruikt om Concept schema's uit meerdere KOSsen aan elkaar te koppelen. In deze fase van het onderzoek is ervoor gekozen om alleen Aquo-lex in SKOS te modelleren. Daardoor worden de klassen ConceptScheme, Collection en OrderedCollection niet gebruikt.

Het totaalbeeld van alle object property relaties (inclusief Thing) is complex en een afbeelding hiervan is slecht leesbaar. Voor de volledigheid is deze wel weergegeven in figuur 7. De gebruikte software heeft goede mogelijkheden om dit totaalbeeld te visualiseren waarbij zoomen en verschuiven mogelijk is. In bijlage 1 is het oorspronkelijke UML ontwerp en het ontwerp van de objectencatalogus naast elkaar weergegeven zodat ze kunnen worden vergeleken.



Figuur 6

De relaties van een concept met de rest van de klassen.



Figuur 7

Alle klassen en object properties van de objectencatalogus.

Tabel 2 is een weergave van alle mogelijke properties (object properties zijn **vetgedrukt**) per klasse. Te zien valt dat alle klassen de eigenschappen van de topklasse 'Thing' overerven vanwege de subklasse relatie die er per definitie in elke ontologie is met 'Thing'. Dit levert enkele recursieve combinaties op die over het algemeen ongewenst zijn (HttpLink heeftHttpLink. Historie heeftHistorie, etc. veroorzaken het z.g. Droste effect). Invullen van ongewenste combinaties door gebruikers zou met een data-invoertool kunnen worden voorkomen.

Tabel 2

*Properties per klasse (object properties zijn **vetgedrukt**, onzinnige properties zijn **rood/cursief** gemarkeerd).*

Domeincatalogus velden	Ontwerp [1] (schuin=abstracte class)	UniqueKey (URI)	Eigenschappen Objectencatalogus	Eigenschappen Ontwerp [1]
Thing	*Iets*	nvt	rdfs:label	naamgeving
			dc:description	dc:description
			heeftHttpLink	httpLink
			dc:creator	dc:creator
			heeftBron	
			dc:source	dc:source
			heeftHistorie	
Historie	Historie	HIS000001	rdfs:label	
			dc:description	
			heeftHttpLink	
			dc:creator	
			heeftBron	
			dc:source	
			heeftHistorie	
			DatumIngangStatus	DatumIngangStatus
			DatumEindeStatus	DatumEindeStatus
			Status	Status
			ActieNemer	ActieNemer
			ObjectVeranderd	ObjectVeranderd
			AttribuutVeranderd	AttribuutVeranderd
			WaardeWas	WaardeWas
			WaardeWerd	WaardeWerd
Autoriteit	Autoriteit	AUT0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			heeftBron	
			dc:source	dc:source
			heeftHistorie	
			beveeltAan	aanbevolenBegrip
Groep	*Groep*	GRP0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			dc:source	dc:source
			heeftBron	
			heeftHistorie	
			heeftKennisgebied	
			heeftWerkproces	

Domeincatalogus velden	Ontwerp [1] (schuin=abstracte class)	UniqueKey (URI)	Eigenschappen Objectencatalogus	Eigenschappen Ontwerp [1]
-	*Object*			
Concept	Term	CON0001	skos:prefLabel + language	naamgeving, Taal
			skos:definition	dc:description
			skos:altLabel	Afkorting
			skos:editorialNote	Toelichting
			synoniem	
			rdfs:label - *niet gebruiken*	
			dc:description - *niet gebruiken*	
			heeftHttpLink	httpLink
			dc:creator	dc:creator
			dc:source	dc:source
			heeftHistorie	
			isAanbevolenDoor	
			heeftBron	
			heeftWerkproces	
			heeftKennisgebied	
			skos:broader	Hogere orde
			skos:narrower	Lagere orde
			alle andere skos properties	
			zwakGerelateerd	Zwak gerelateerd
			sterkGerelateerd	Sterk gerelateerd
Bron	*Bron*	BRN0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			dc:source	dc:source
			heeftBron	
			bepaalt	
			heeftHistorie	
Graphic Subklasse van Bron ('is-a Bron)	Graphic	GRA0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			heeftBron	
			dc:source	dc:source
			heeftHistorie	
HttpLink Subklasse van Bron ('is-a Bron)	HttpLink	HTL0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			heeftBron	
			dc:source	dc:source
			heeftHistorie	
LitRef Subklasse van Bron ('is-a Bron)	LitRef	LIT0001	rdfs:label	naamgeving
			dc:description	dc:description
			dc:creator	dc:creator
			heeftHttpLink	httpLink
			heeftBron	
			dc:source	dc:source
			heeftHistorie	

Domeincatalogus velden	Ontwerp [1] (schuin=abstracte class)	UniqueKey (URI)	Eigenschappen Objectencatalogus	Eigenschappen Ontwerp [1]
AquoBron Subklasse van Bron ('is-a Bron)	-	ABR0001	rdfs:label dc:description dc:creator **heeftHttpLink** ***heeftBron*** dc:source **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source
Kennisgebied	Kennisgebied	KGB0001	rdfs:label dc:description dc:creator **heeftHttpLink** **heeftBron** dc:source **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source
Werkproces	Werkproces	WER0001	rdfs:label dc:description dc:creator **heeftHttpLink** **heeftBron** dc:source **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source
AquoDomeinHoedanigheid	AquoDomeinHoedanigheid	HOE0001	rdfs:label dc:description dc:creator **heeftHttpLink** dc:source **heeftConcept** **heeftBron** Code **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source code
AquoDomeinEenheid	AquoDomeinEenheid	EEN0001	rdfs:label dc:description dc:creator **heeftHttpLink** dc:source **heeftBron** Code **heeftConcept** **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source code
EntiteitKlasse	EntiteitKlasse	ENT0001	rdfs:label dc:description dc:creator **heeftHttpLink** **heeftBron** dc:source **heeftConcept** Code visualisatie **heeftKenmerk** **heeftHistorie**	naamgeving dc:description dc:creator httpLink dc:source code visualisatie

Domeincatalogus velden	Ontwerp [1] (schuin=abstracte class)	UniqueKey (URI)	Eigenschappen Objectencatalogus	Eigenschappen Ontwerp [1]
Kenmerk	Kenmerk	KEN0001	rdfs:label	naamgeving
			Dc:description	dc:description
			Dc:creator	dc:creator
			heeftHttpLink	httpLink
			Dc:source	dc:source
			heeftBron	
			code	code
			visualisatie	visualisatie
			heeftEenheid	
			heeftHoedanigheid	
			heeftConcept	
			heeftHistorie	

In tabel 3 zijn de properties weergegeven die zijn gebruikt. Verschillende properties komen uit bestaande namespaces (dublin core dc, rdf en rdfs, skos). Voor de objectencatalogus zijn eigen properties gedefinieerd in de 'catalogus' namespace.

Tabel 3

Alle gebruikte properties en hun namespace.

Namespace	Property	Type	Triple
dc	creator	String	Datatype property bij Thing
	description	String	Datatype property bij Thing
	source	String	Datatype property bij Thing
rdfs	label	String	Datatype property bij rdfs:Resource
rdf	type (de functie hiervan is het weergeven van de klasse waartoe het object behoort)	rdf:Property	Datatype property bij rdfs:Resource
skos	skos:prefLabel	String	Datatype property bij Concept
	skos:altLabel	String	Datatype property bij Concept
	skos:editorialNote	String	Datatype property bij Concept
	skos:definition	String	Datatype property bij Concept
	skos:broader	Object	Concept skos:broader Concept
	skos:narrower	Object	Concept skos:narrower Concept
catalogus	bepaalt	Object	Bron bepaalt Concept
	beveeltAan	Object	Autoriteit beveeltAan Concept (inverse van isAanbevolenDoor)
	heeftBron	Object	Concept heeftBron Bron
	heeftConcept	Object	AquoDomeinEenheid heeftConcept Concept; AquoDomeinHoedanigheid heeftConcept Concept; Kenmerk heeftConcept Concept; EntiteitKlasse heeftConcept Concept
	heeftEenheid	Object	Kenmerk heeftEenheid AquoDomeinEenheid

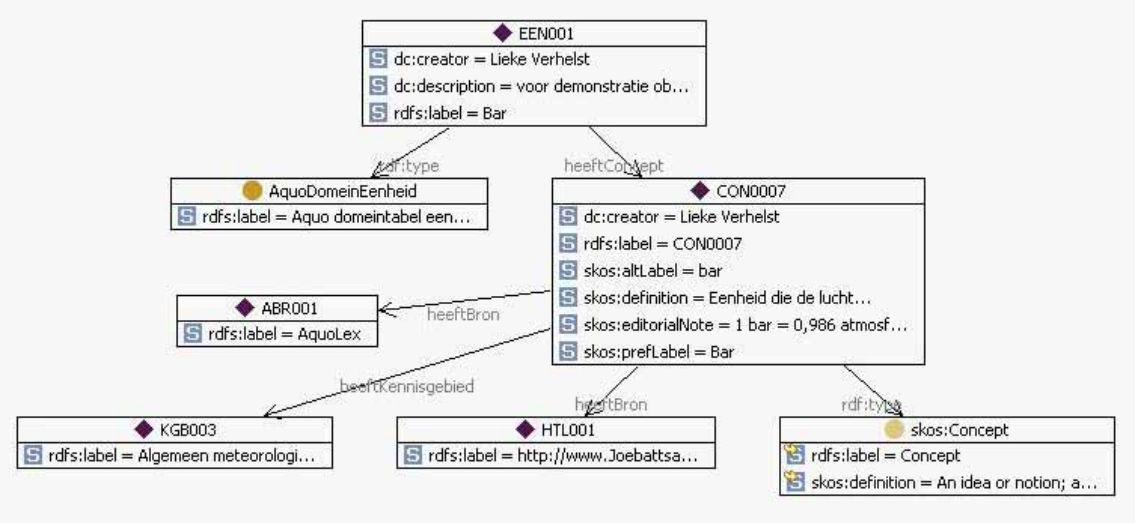
Namespace	Property	Type	Triple
	heeftGroep	Object	Kennisgebied heeftGroep Groep Werkproces heeftGroep Groep Concept heefGroep Groep
	heeftHistorie	Object	Thing heeftHistorie Historie
	heeftHoedanigheid	Object	Kenmerk heeftHoedanigheid AquoDomeinHoedanigheid
	heeftHttpLink	Object	Thing heeftHttpLink HttpLink
	heeftKenmerk	Object	EntiteitKlasse heeftKenmerk Kenmerk
	heeftKennisgebied	Object	Groep heeftKennisgebied Kennisgebied; Concept heeftKennisgebied Kennisgebied
	heeftWerkproces	Object	Groep heeftWerkproces Werkproces; Concept heeftWerkproces Werkproces
	isAanbevolenDoor	Object	Concept isAanbevolenDoor Autoriteit (inverse van beveeltAan)
	sterkGerelateerd	Object	Concept sterkGerelateerd Concept
	zwakGerelateerd	Object	Concept zwakGerelateerd Concept
	actienemer	string	Datatype property bij Historie
	attribuutVeranderd	String	Datatype property bij Historie
	code	String	Datatype property bij EntiteitKlasse, Kenmerk, AquoDomeinEenheid, AquoDomeinHoedanigheid
	datumEindeStatus	Date	Datatype property bij Historie
	datumIngangStatus	Date	Datatype property bij Historie
	objectVeranderd	String	Datatype property bij Historie
	status	String	Datatype property bij Historie
	synoniem	String	Datatype property bij Concept
	visualisatie	String	Dataproperty bij EntiteitKlasse en Kenmerk
	waardeWas	String	Datatype property bij Historie
	waardeWerd	String	Datatype property bij Historie

Samenvattend zijn in de objectencatalogus de volgende klassen gebruikt:
Thing, Concept, Historie, Autoriteit, Groep, Werkproces, Kennisgebied, Bron, Graphic, HttpLink, LitRef, AquoBron, EntiteitKlasse, Kenmerk, AquoDomeinEenheid, AquoDomeinHoedanigheid.
In de ontologie editor zien we meer klassen dan de bovenstaande, dat komt hierdoor: naast de Concept klasse zijn door het gebruiken van het SKOS schema nog de klassen Collection, OrderedCollection en ConceptScheme aanwezig. Deze worden niet gebruikt. De Thing klasse is altijd aanwezig in een ontologie.

In de volgende paragraaf zijn enkele voorbeelden gegeven waardoor de werking van de objectencatalogus wordt verduidelijkt.

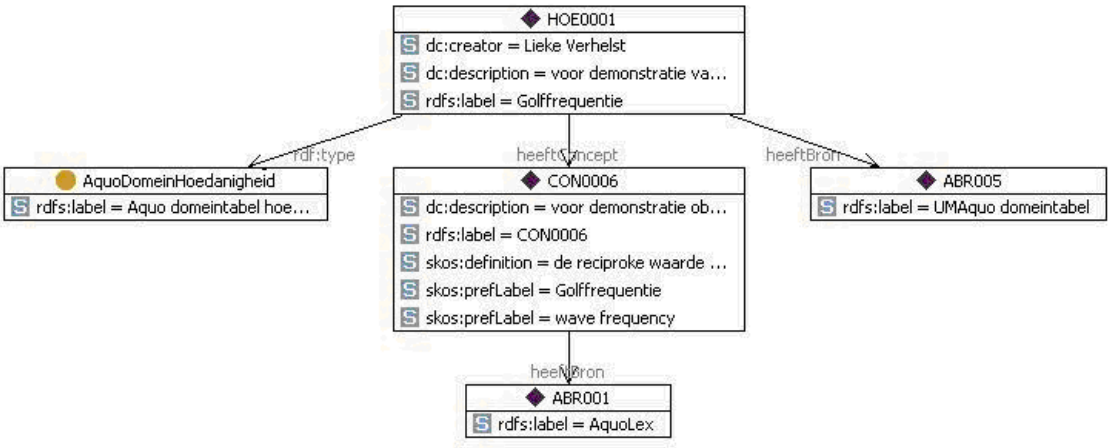
4.3 Praktijkvoorbeelden

Onderstaande afbeeldingen geven enkele uitwerkingen van hoe objecten van respectievelijk de domeintabellen en LM Aquo en IMWA/UM Aquo zijn gekoppeld aan Aquo-lex.
In de huidige versie van de standaarden is het begrip 'bar' opgenomen in Aquo-lex met hierin ook de gegevens voor het kennisgebied en de referentie link. In de LM Aquo domeintabel voor spanningsklasse is de eenheid 'bar' opgenomen. Er is geen automatische koppeling tussen Aquo-lex en de LM Aquo domeintabel. In de objectencatalogus ziet het er als volgt uit:



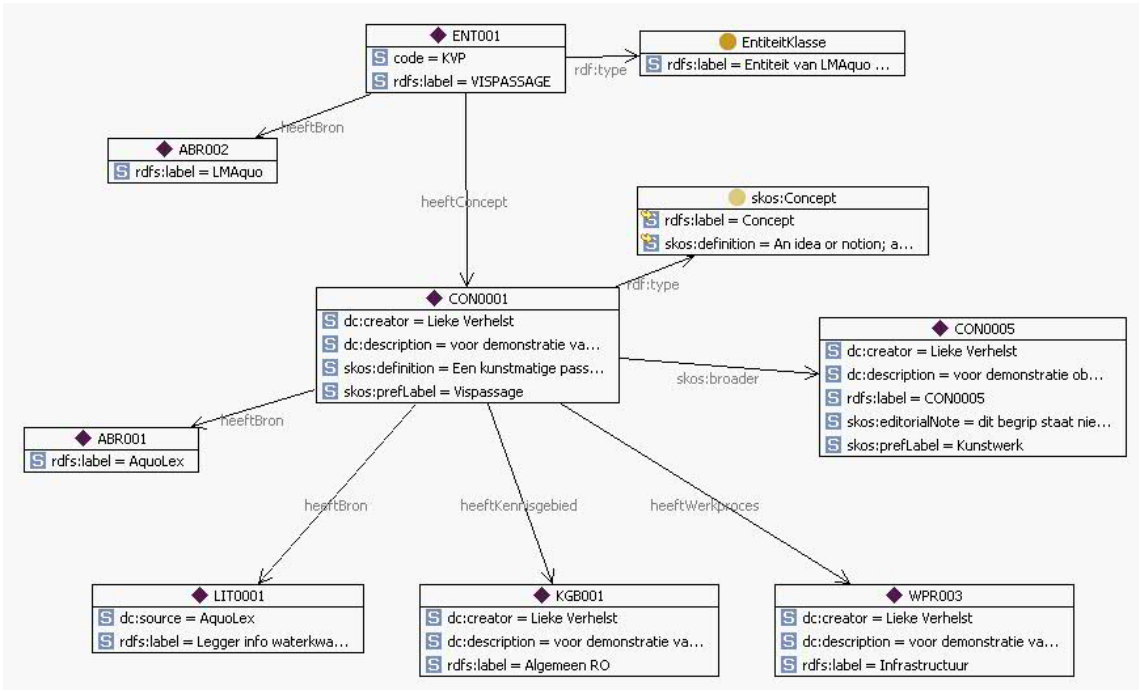
Figuur 8
De eenheid 'Bar' met de gegevens uit Aquo-lex.

De hoedanigheid 'Golffrequentie' komt momenteel voor in een UM Aquo domeintabel (XSD bestand) en in Aquo-lex. Tussen deze bestanden is geen automatische koppeling. Bij de objectencatalogus is de koppeling als volgt gerealiseerd:



Figuur 9
De hoedanigheid 'Golffrequentie' gekoppeld aan Aquo-lex.

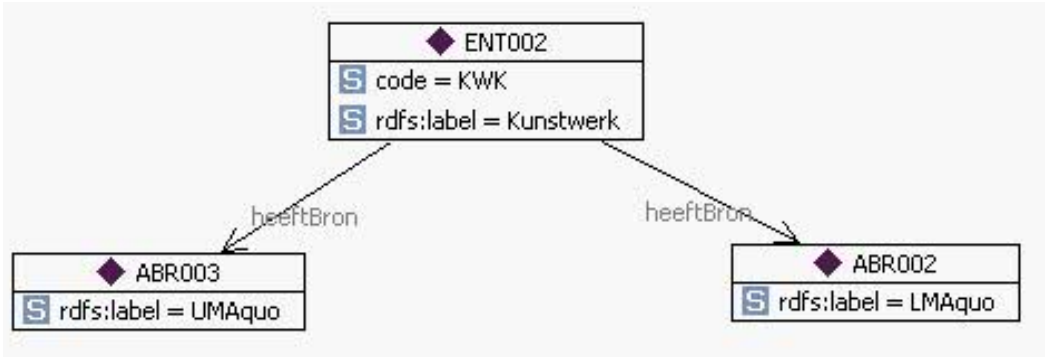
Vispassage is een Entiteit in LM Aquo. Het is ook een begrip in Aquo-lex. Deze bestanden zijn niet automatisch gekoppeld. In de objectencatalogus wordt de relatie tussen deze items als volgt gemodelleerd:



Figuur 10

De koppeling van LM Aquo entiteit Vispassage aan begrip Vispassage in Aquo-lex.

Van het object 'Kunstwerk' bestaat een entiteit in LM Aquo en een klasse in IMWA/UM Aquo, en een begrip in Aquo-lex (figuur 11).

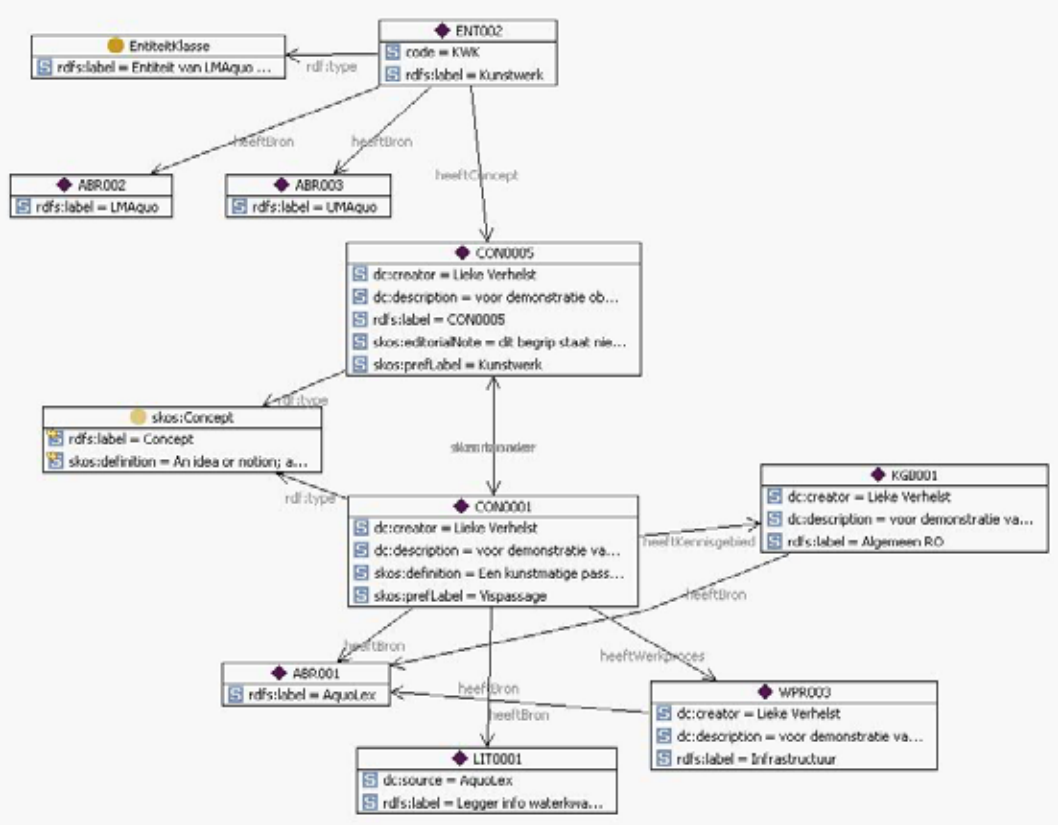


Figuur 11

'Kunstwerk' komt voor in IMWA/UM Aquo en LM Aquo.

'Concept' is het centrale punt in het semantische model van de objectencatalogus. Op deze manier is 'Kunstwerk' via een 'broader' concept gekoppeld aan 'Vispassage'. Dit is weergegeven in figuur 12. In deze figuur is ook te zien dat de vakterm 'kunstwerk' uit LM Aquo en UM Aquo gekoppeld is aan Aquo-lex.

NB: er is geen 'broader' of 'narrower' relatie vastgelegd vanuit 'EntiteitKlasse' alleen vanuit 'Concept' (zie ook paragraaf 4.2). In de huidige Aquo standaarden zijn de 'broader' en 'narrower' relaties weergegeven in LM Aquo door het aangeven van de aanwezigheid van een superentiteit.



Figuur 12

De entiteit 'Kunstwerk' uit LM Aquo met de gekoppelde gegevens uit Aquo-lex (inclusief narrower relatie 'Vispassage').

4.3.1 Cardinaliteit en constraints

In de huidige versie van de objectencatalogus zitten geen constraints en cardinaliteitsregels anders dan de opgelegde datatypen. Het datatype wordt wel bij invoer gecontroleerd: bijvoorbeeld een stringwaarde kan niet worden ingevuld in een integerveld. Cardinaliteit wordt bij sommige ontologie editors bij data-invoer gecheckt door het minimaal invullen van een bepaald aantal waarden (bij vroegere versies van Protégé) soms moet de editor hiervoor worden geconfigureerd (Topbraid Composer, Protégé 4). De regels voor cardinaliteit in de opgeleverde objectencatalogus zijn niet ingevuld. Daardoor is cardinaliteit onbepaald (er mag een onbeperkt aantal waarden worden ingevuld en ook 0 waarden.)

De opgelegde invoer en cardinaliteitsconstraints uit het oorspronkelijke ontwerp [1] kunnen in OWL worden vastgelegd. In de huidige versie van de objectcatalogus zijn deze regels nog niet aanwezig. De volgende versie van de objectencatalogus zal deze cardinaliteitsregels wel moeten hebben.

De consistency constraints worden zoals gezegd achteraf gevalideerd. De gevolgen hiervoor worden besproken in paragraaf 4.5.1 (constraints).

4.4 Het laden van de bestaande data

De data die in de objectencatalogus geladen moet worden is afkomstig van verschillende bronnen met een verschillend formaat. Deze bestanden zijn allemaal te vinden op de website van IDsW. Het is bij ons onderzoek wel nodig geweest om van de IDsW helpdesk aanwijzingen te krijgen welk bronbestand waar te vinden was. Aquo-lex is een Excel bestand, de domeintabellen van UM Aquo komen voor dit project uit de XSD (er is ook een txt/csv versie beschikbaar), de gegevens in LM Aquo bevinden zich in Oracle tabellen en in Excel (domeintabellen) en UM Aquo/IMWA is een UML model.

Om al deze bronnen samen te brengen in een enkel XML bestand van formaat OWL is gekozen voor de import en conversie tool FME van SAFE software inc[1]. Met FME kunnen verschillende bronnen worden ingelezen, waarna de data gerouteerd kan worden naar output bestanden. 'Onderweg' kunnen conversies worden uitgevoerd, zoals nummeringen, aaneenschakelen en het toevoegen van strings. De input-output mapping kan eenvoudig in een interactieve 'workbench' worden aangepast. Dit maakt het een bruikbare tool voor proof of concept trajecten zoals dit project.

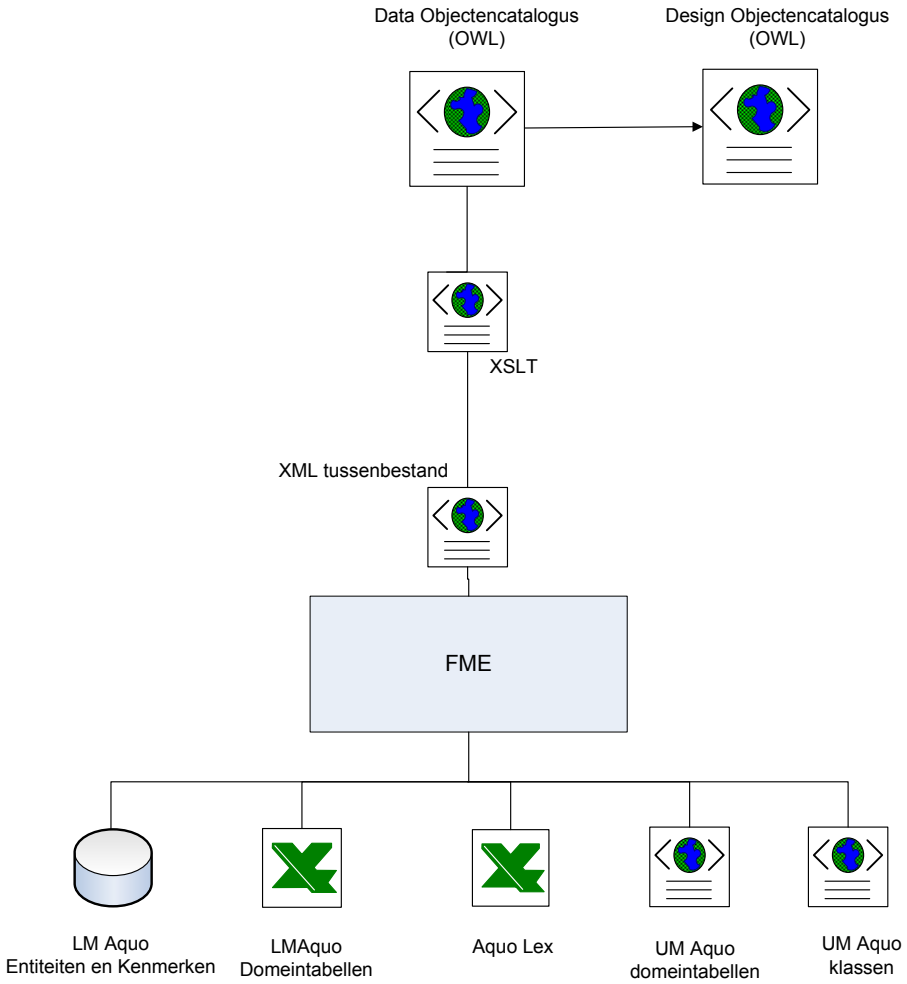
Het is eenvoudig gebleken om op deze manier Excel en Oracle tabellen in te lezen. Echter door historisch gegroeide organisatorische redenen binnen IDsW zijn de Oracle tabellen van LM Aquo niet toegankelijk. Er is in dit onderzoek gewerkt met een set tabellen die niet voorzien waren van primary key-foreign key relaties. Als gevolg daarvan zijn enkel lijsten van Entiteiten en Kenmerken ingelezen, zonder de relatie tussen die twee. Deze relatie is wel beschikbaar in een HTML export van de LM Aquo database, zodat door het ontwerpen van een parser de relaties toch kunnen worden afgeleid. In verband met de beperkte scope van dit onderzoek is geen parser geschreven om deze actie ook daadwerkelijk te kunnen uitvoeren.

De LM Aquo domeintabellen voor Eenheid en Hoedanigheid zijn beschikbaar op de website in Excel en deze zijn ingelezen. De relatie tussen de domeinen en de kenmerken is niet beschikbaar in het betreffende Excel bestand. In verband met de beperkte scope is deze relatie niet vastgelegd in het objectencatalogus OWL importbestand.

De Klassen in UM Aquo (IMWA) zijn beschikbaar in een UML bestand. Het is mogelijk UML te converteren naar XML en van daaruit via XSLT of import in Excel de gegevens van de klassen op te halen. In verband met de beperkte scope van dit onderzoek is deze conversie niet uitgevoerd. De lijst met klassen uit IMWA is ingelezen.

FME is niet in staat om een geformatteerd XML bestand op te leveren. Het XML formaat wat door FME wordt opgebouwd bij een dataconversie bevat een lange rij tags (bijvoorbeeld voor het veld prefLabel <fme:prefLabel> </fme:prefLabel>). Met een XSLT conversie is dit XML bestand omgezet naar een OWL bestand. Het totale import proces is afgebeeld in figuur 13.

[1] http://www.safe.com/



Figuur 13

De conversie van de verschillende bronnen naar het OWL bestand van de objectencatalogus.

Het aldus verkregen OWL databestand kan worden geïmporteerd in het ontwerp van de objectencatalogus. In de ontology header van de objectencatalogus wordt dit vastgelegd (zie tekst in **vetgedrukt**):

```
<owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://purl.org/dc/elements/1.1/"/>
   <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Objectencatalogus van IDsW gemaakt door Alterra, dec 2009</owl:versionInfo>
      <owl:imports rdf:resource="http://www.w3.org/2004/02/skos/core"/>
      <owl:imports rdf:resource="/owl/2009/data/objcatindiv"/>
   </owl:Ontology>
```

Op deze manier is de data gescheiden van het ontwerp van de objectencatalogus.

Na de import ontstaat er een bestand dat data bevat van de domeintabellen, IMWA/UM Aquo LM Aquo en Aquo-lex. Als dit bestand verder als bronbestand wordt gebruikt waarin wijzigingen worden bijgehouden kunnen dan in principe dan delen van deze oorspronkelijke bronnen komen te vervallen. Dit zou in een vervolgonderzoek kunnen worden uitgewerkt.

Indien de objectencatalogus in de toekomst als overkoepelende bron zal worden gebruikt voor (de overgebleven delen van) UM Aquo en LM Aquo moet een conversie terug naar deze standaarden worden ontworpen:
– voor LM Aquo zou dat in principe zijn: OWL naar database tabellen;
– voor de UM Aquo UML deel van UM Aquo: OWL naar UML.

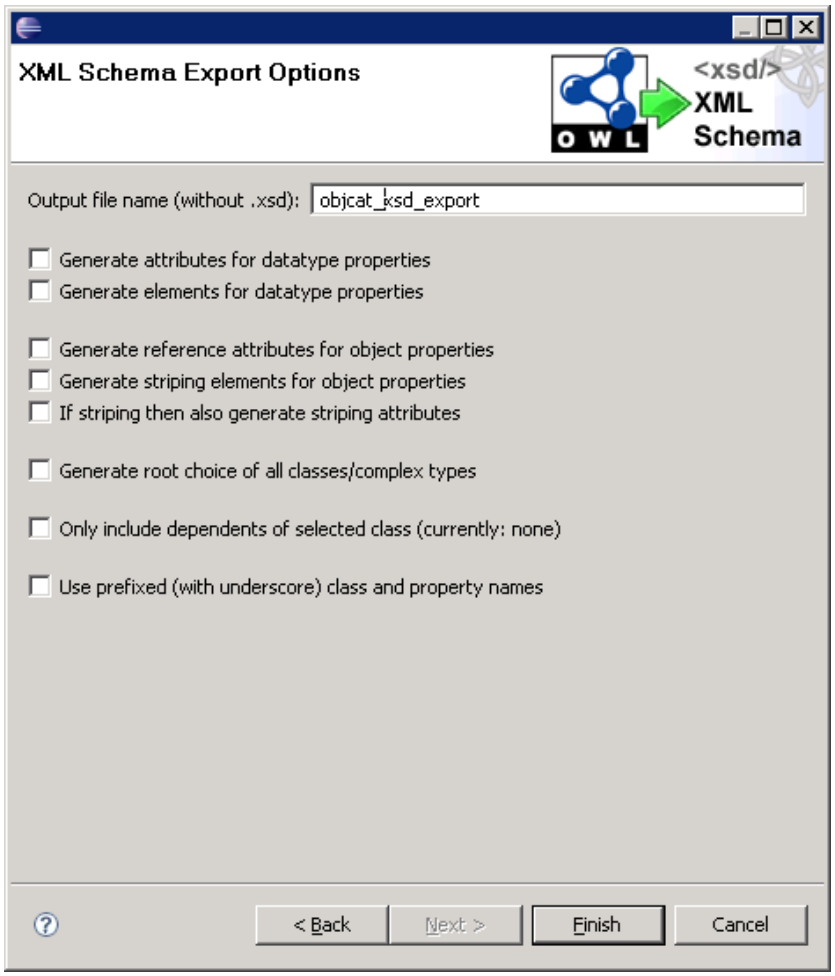
De conversie van OWL naar de oorspronkelijke formaten is in dit onderzoek onderzocht. De koppeling van OWL naar database bronnen is eenvoudig te doen. De gebruikte ontologie editors hebben plugins[2] die koppelingen met databases mogelijk maken, en een tool als FME is ook zeer bruikbaar. Op het moment dat de conversiestrategie vastligt kan dit door middel van scripting worden geregeld, zodat de aanschaf van FME niet noodzakelijk is.

De koppeling van OWL naar UML (of XMI) of omgekeerd is complexer. Tests met verschillende tools zoals Topbraid Composer en de Altova MissionKit hebben aangetoond dat postprocessing van de geïmporteerde of geëxporteerde data naar het gewenste formaat nodig is. Bijvoorbeeld (import naar OWL): een XSD wordt door Topbraid Composer geconverteerd naar een reeks klassen (bijvoorbeeld annotatie, enummeratie) met daarvan individuals. Dit moet worden geconverteerd naar de objectcatalogus klassen en individuals.
Met de Altova MissionKit kan een XSLT bestand worden ontworpen die voor de conversie van de XSDs naar OWL zorgt. Het UML bestand van Enterprise Architect kan worden geëxporteerd naar XML bestanden die met XSLT kunnen worden omgevormd naar OWL. Er zijn ook UML naar OWL conversies gebaseerd op XMI naar OWL via XSLT beschikbaar op het internet.[3]
Voor conversie van OWL naar XSD (export uit OWL) is mogelijk in Topbraid Composer (zie figuur 14). Hoe dit uitpakt in de praktijk met de XSDs van UM Aquo is in dit onderzoek nog niet getest. Voor wat betreft de conversie van OWL naar UML (klassen en enumeraties), hiervoor zijn op het internet conversietools beschikbaar.[4] Deze zijn in dit onderzoek niet getest.

[2] Topbraidcomposer Maestro en Standaard Editie is voorzien van koppeling met RDF databases, zie http://www.topquadrant.com/products/TB_Composer.html
Voor protege bestaan er verschillende plugins die data importeren en exporteren.

[3] http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/

[4] http://biopaxwiki.org/cgi-bin/moin.cgi/Converting_OWL_To_UML



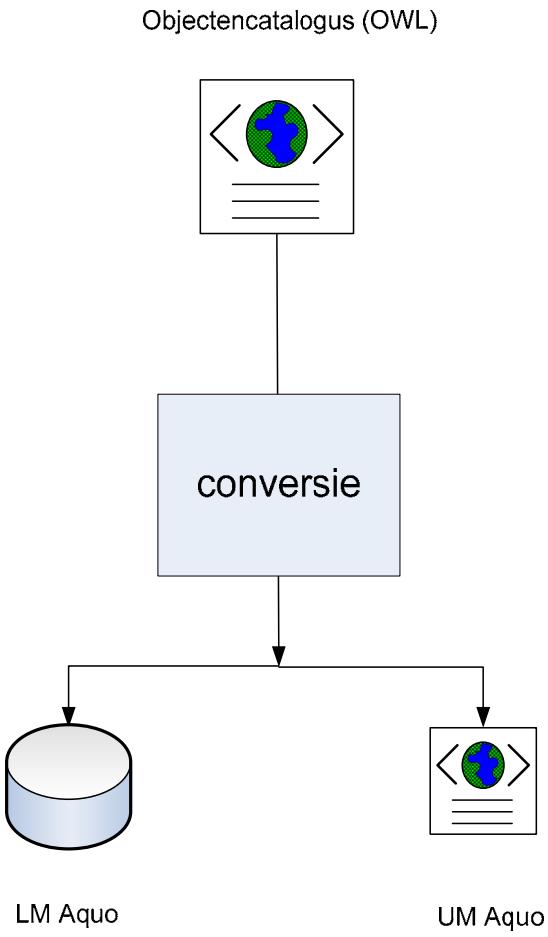
Figuur 14

Het specificatiescherm van Topbraid Composer voor een OWL export naar XSD.

Samenvattend zijn de volgende data conversies te onderscheiden:

1. eenmalige import van overkoepelende gegevens van bestaande Aquo Standaarden in de objectencatalogus (figuur 13);
2. ontwerp van koppeling van objectencatalogus (bron) naar overblijvende Aquo Standaarden (mogelijke situatie geschetst in figuur 15).

Wat betreft het omgaan met de data in de objectencatalogus is in het onderzoek verder aan het licht gekomen dat het door elkaar gebruiken van ontologie editors op een OWL bestand niet aan te raden is. De verschillende ontologie editors schrijven OWL in een andere XML opmaak weg.



Figuur 15
Mogelijke toekomstige situatie.

4.5 Het beheer van de objecten catalogus

In paragraaf 3.2 van [1] worden de eisen beschreven die zijn gesteld aan het beheer van de objectencatalogus. Hierbij is aandacht gegeven aan de invoer van de data (aanmaken, wijzigen, verwijderen, validatie), aan het bewaren van de historie van de data, aan het ontsluiten van de data, en aan de beschikbaarheid en performance van de applicaties die zorg dragen voor ontsluiting via het internet.
In het onderzoek is naar voren gekomen dat, wanneer de objectencatalogus als primair bronbestand wordt gebruikt, aan deze lijst moet worden toegevoegd het bewaken van de juiste koppeling tussen de objectencatalogus en de andere Aquo standaarden (zie paragraaf 4.4).
Ten tweede moet de integriteit van de data in de objectencatalogus worden bewaakt. Hiermee wordt bedoeld dat de data niet alleen correct wordt ingevoerd (met behulp van invoervalidatie), maar dat ook bij opslag de correctheid van de data gegarandeerd blijft. Te denken valt aan het corrupt raken van het OWL bestand. De objectencatalogus bestaat uit twee OWL bestanden: het design en de data. Deze bestanden zijn tekstbestanden van het formaat OWL-XML. De integriteit van deze bestanden kan niet worden gegarandeerd als er geen mechanisme aan wordt toegevoegd in de vorm van een proces of techniek. Bij een proces valt de denken aan het afdwingen dat één persoon of identiteit toegang heeft tot de OWL bestanden en dat na elke wijziging altijd een back-up wordt gemaakt. Een goede technische oplossing zou zijn een onderliggende database die de data integriteit en -veiligheid garandeert. De door ons onderzochte ontologie editors kunnen verbinding maken met

backenddatabases. Hoe deze kunnen worden gebruikt om de integriteit van de data te bewaren wordt besproken in paragraaf 4.5.1.

4.5.1 Tools

We maken in deze paragraaf onderscheid tussen tools voor het beheren van de data enerzijds en voor het beheer van het ontwerp van de objectencatalogus anderzijds.

Data beheer
Er zijn verschillende redenen te noemen die er voor pleiten gebruik te maken van een backenddatabase achter de objectencatalogus. Al genoemd is het bewaken van de data-integriteit. Een tweede is dat het beheren van een groot OWL bestand wordt beperkt door het aanwezige computergeheugen en -verwerkingscapaciteit. Een achterliggende database waarin de RDF triples worden opgeslagen kan een oplossing hiervoor zijn.

De objectencatalogus is een OWL bestand dat in principe met elke teksteditor kan worden gelezen. Een XML editor, zoals XMLSpy of XMLStudio heeft als toegevoegde waarde dat de tekst wordt opgemaakt met kleuren en tabs. Een ontologie editor zoals bijvoorbeeld Protégé of Topbraid Composer geeft de klassen, relaties en de constraints in duidelijke schema's weer. In onderstaand overzicht zijn de ontologie editors Protégé en Topbraid Composer met elkaar vergeleken ten opzichte van de aan de objectencatalogus gestelde eisen [1]. Te zien valt dat beide editors niet helemaal voldoen aan deze gestelde eisen.

Tabel 4

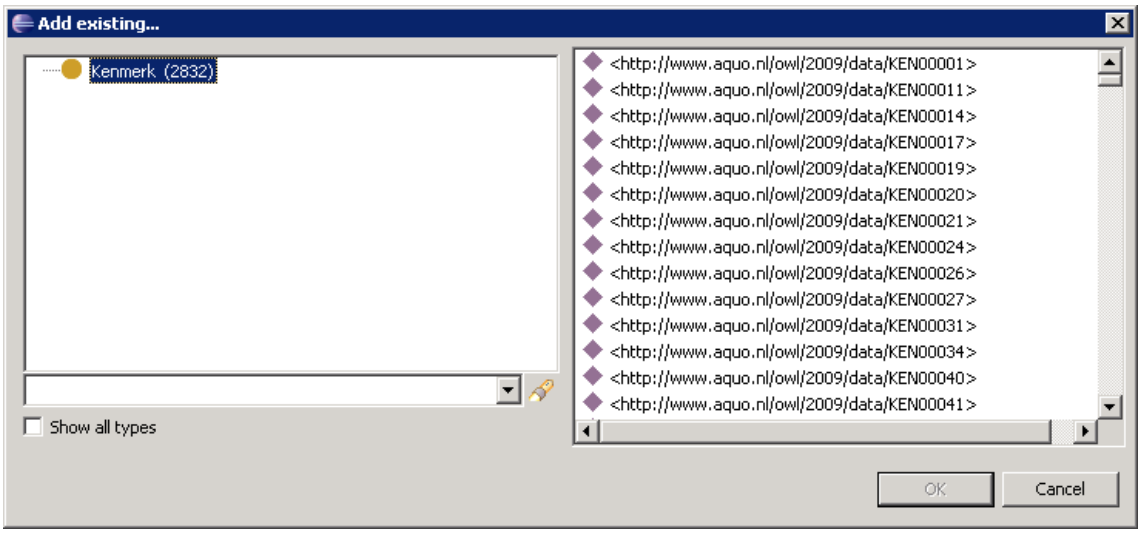
Vergelijkende tabel van ontologie editors.

		Topbraid composer (Free edition)	Protégé V4 met SKOS editor plugin[5]
Aanmaken		Ja, maar niet erg overzichtelijk	Ja, maar niet erg overzichtelijk
Importeren		Ja, meerdere typen. Hierna batch conversie via TBC editor mogelijk	Meerdere import plugins beschikbaar
Wijzigen		Ja	Ja
Verwijderen		Ja	Ja
Validatie			
	Unieke sleutelwaarde	Ja	Ja
	Conform kolomdefinitie	Ja	Ja
	Check op correct datatype (integer, string, etc.)	Ja	Ja
	Check op verplicht veld (cardinaliteit)[6]	Ja, na specifieke configuratie van TBC	Ja, met additionele plugins
	Relaties	Ja, via unieke sleutelwaarde (URI)	Ja, via unieke sleutelwaarde (URI)
	Datum is geldig	Ja	Ja
	Tijd is geldig	Ja	Ja
Beheren van lijsten		Ja, maar tonen van lijsten niet overzichtelijk	Ja, maar tonen van lijsten niet overzichtelijk
Automatisch tonen inverse relatie		Ja, na reasonen	Ja, na reasonen

Voor het invoeren van grote hoeveelheden data zijn beide editors niet geschikt, omdat de schermen te complex zijn en mogelijkheden bieden voor foutieve invoer. Bijvoorbeeld: voor het maken van een object type koppeling tussen individuals worden beschikbare individuals getoond, maar alleen met hun unieke sleutel (zie figuur 16 en figuur 17).

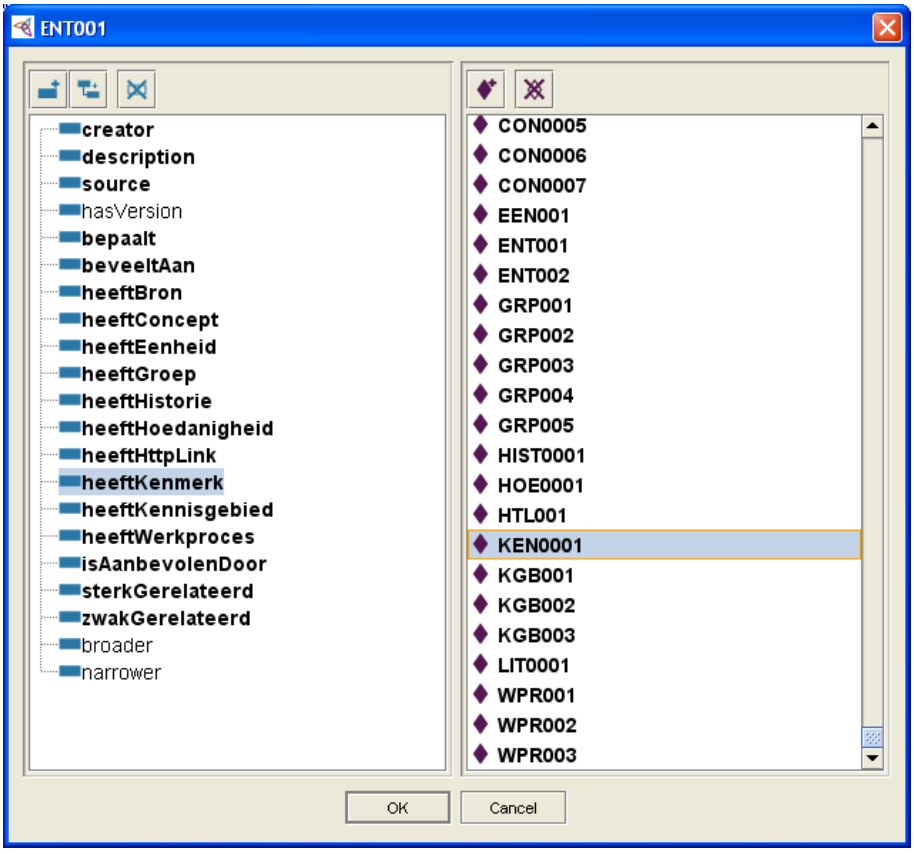
[5] http://code.google.com/p/skoseditor/

[6] Dit is vastgesteld in de betreffende documentatie, maar door ons niet getest.



Figuur 16

Invoeren van object property Kenmerk aan entiteit ENT001 in Topbraid Composer.



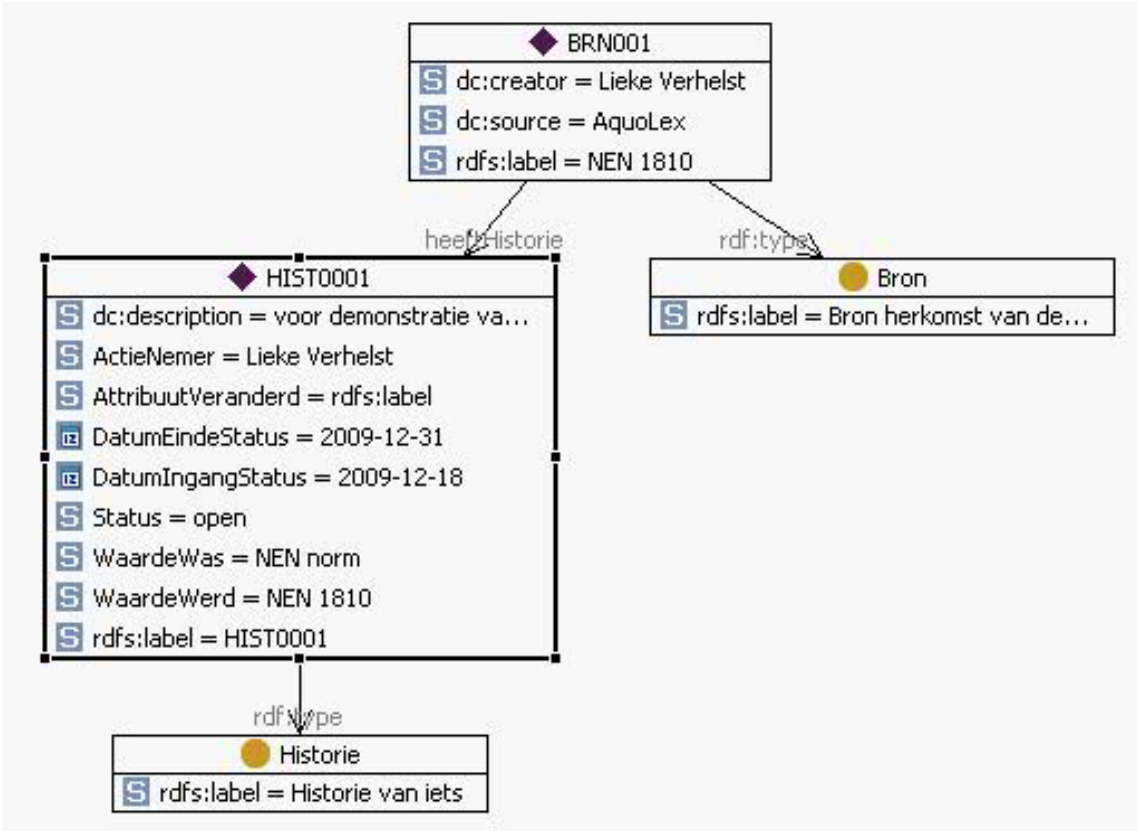
Figuur 17

Invoeren van object property aan entiteit ENT001 in Protégé V4.

Ook is het genereren van overzichten niet voldoende ondersteund. In Topbraid Composer worden alleen de eerste 1000 individuals getoond, waardoor het lijkt alsof er individuals missen. In Protégé worden wel alle individuals getoond in een lange lijst unieke sleutels. Topbraid Composer biedt de mogelijkheid om kolomdefinities in overzichten aan te passen.

Beheer Historie

Voor het bewaken van historie is in het huidige model een klasse Historie aangemaakt met een reeks attributen. Om de historie van een item in de objectencatalogus te documenteren zal een individual van de historieklasse moeten worden gekoppeld met de eigenschap 'heeftHistorie' aan het item. Dit zou in de praktijk als volgt moeten werken: als er een item wordt gewijzigd wordt eerst een nieuw individual in de historieklasse aangemaakt waarin alle eigenschappen staan die te maken hebben met de wijziging (zoals waardeWas, waardeWerd, actienemer, etc.). Hierna moet deze individual van de Historieklasse via de heeftHistorie property aan het item worden gekoppeld. Dit is grafisch weergegeven in figuur 18. Deze manier van werken werkt het maken van fouten in de hand omdat het bijhouden van historische wijzigingen wordt overgelaten aan degene die de data wijzigt in plaats van dat dit automatisch in het systeem wordt geregeld. Bovendien is het bijzonder tijdrovend.



Figuur 18

Het koppelen van historiegegevens aan een gewijzigd object.

Ons advies is te kiezen voor een onderliggende database die de historie van de objecten bewaakt en waar mogelijk geautomatiseerd invult.

Beheer Constraints
De objectencatalogus bevat in zijn huidige gedaante geen constraints, behalve de opgelegde datatypen. Een dergelijk databestand laat voor een praktijksituatie teveel interpretatiemogelijkheden aan de gebruiker. Onjuist gebruik van deze ontologie kan de betekenis van het datamodel veranderen. Een voorbeeld is al gegeven in een eerdere paragraaf:

EntiteitKlasse heeftEenheid AquoDomeinEenheid is geldig
Concept heeftEenheid AquoDomeinEenheid is ongeldig

Hoewel de property *heeftEenheid* een opgelegd domein heeft (namelijk *Kenmerk*) is het mogelijk om in het formulier van Topbraid Composer of Protégé de property *heeftEenheid* toe te voegen aan Concept klasse. Het gevolg hiervan is dat na consistency checking met een reasoner wordt geconcludeerd dat de oorspronkelijke Concept individual niet alleen van het type Concept is maar ook van het type Kenmerk vanwege de relatie heeftEenheid, die immers als domein Kenmerk heeft. Met andere woorden: het open karakter van een OWL bestand brengt met zich mee dat goed moet worden nagedacht over constraints (integriteitsregels) en communicatie over gebruik voordat het kan worden gepubliceerd aan derden.
Constraintregels worden in een ontologie-editor tool vastgelegd bij een klasse of bij een property (zie hoofdstuk 3).

Het maken van een OWL bestand met het juiste formaat is een te nauwkeurige taak om met een tekst- of XML editor te doen. De ontologie-editors Protégé en Topbraid Composer zijn elkaars concurrent hierin. Topbraid Composer is in principe een commercieel product, maar er is ook een gratis versie. Protégé is een open source editor gemaakt door de universiteit van Stanford. Beide editors hebben globaal gesproken dezelfde functionaliteit. Topbraid composer is gebruikersvriendelijker en overzichtelijker.
Topbraid Composer onderscheidt zich verder van Protégé door duidelijke user interface, betere stabiliteit en een aparte tab voor foutmeldingen. In dit onderzoek is daarom voornamelijk gewerkt met de Topbraid Composer Free Edition. De Standaard versie bevat een goede mogelijkheid om ontologieën grafisch weer te geven. Deze is voor de afbeeldingen in het rapport gebruikt.

4.6 Het ontsluiten van een OWL op het internet

Tijdens het onderzoek naar het ontsluiten van een OWL file via een webapplicatie en een webservice zijn verschillende mogelijkheden uitgewerkt. Dit is gedaan omdat ingeslagen wegen onverwachte uitdagingen met zich mee brachten.
Voor het bouwen van de webapplicaties hebben we drie verschillende routes genomen:
1. het bouwen van een website gebaseerd op JSP met Java code gegenereerd door Protégé 3.4;
2. het bouwen van een website gebaseerd op JSP met Java code gegenereerd door een open source plugin in combinatie met Topbraid Composer;
3. conversie van het OWL XML bestand naar HTML via XSLT.

De drie methodes zijn ten opzichte van elkaar geëvalueerd waarbij is gelet op ontwikkeltijd, software kosten en flexibiliteit bij doorvoeren van wijzigingen.

Ad 1.
Protégé 3.4 heeft de mogelijkheid om Java Schema classes te genereren op basis van een OWL file dat in Protégé is geopend. In Protégé Versie 4 is dit niet meer aanwezig, vandaar dat deze versie van Protégé is gebruikt ondanks de verouderde status van 3.4. De gegenereerde code maakt gebruik van de Protégé OWL Application Programming Interface (API, oftewel bibliotheek) om het OWL bestand in te lezen en te parsen.

Van deze functionaliteit hebben we gebruik gemaakt om Java classes te genereren die als basis dienen voor een JSP applicatie.
Hierbij zijn we op de volgende praktische problemen gestuit:

- bij wijziging van de structuur van de OWL moet opnieuw Java code worden gegenereerd;
- niet alle ingevulde data kan worden verwerkt: speciale tekens kunnen problemen geven;
- als in het gebruikte OWL bestand termen worden gebruikt die in Java standaard klassenamen zijn, dan moeten er speciale maatregelen worden genomen om te zorgen dat er geen code ontstaat vol met ambigue referenties;
- om de Protégé OWL API werkend te krijgen moesten we aan onverwachte requirements voldoen; zo moesten we extra geheugen reserveren en bleek ons na lang zoeken ook dat het nodig was om met de hand een *.repository file aan te maken die normaal door Protégé wordt aangemaakt bij het openen van het gelijknamige OWL bestand in de Protégé editor; bij het aanroepen van alleen de API vanuit de gegenereerde Java code gebeurde dat niet vanzelf;
- de Protégé OWL API bleek erg onoverzichtelijk en slechts beperkt gedocumenteerd, hierdoor kan het ontwikkelen van de JSP code relatief veel tijd kosten.

In de loop van het project is het ons duidelijk geworden dat het gebruik van de Protégé OWL API voor het inlezen van het OWL bestand af te raden is. De reden hiervoor is dat de twee onderzochte ontologie-editors de data op enigszins verschillende manieren opslaan in het OWL bestand. De in dit project ontwikkelde objectencatalogus OWL file is gemaakt met Topbraid Composer, en bij gebruik van de Protégé OWL API om datzelfde bestand in te lezen kan het zijn dat bepaalde delen van het bestand worden genegeerd en uiteindelijk dus niet worden weergegeven. Het gebruik van de Protégé OWL API om in datzelfde bestand te schrijven is zelfs riskant: in theorie kan Topbraid Composer het resulterende bestand dan als corrupt gaan aanmerken. Om deze reden zijn we gaandeweg het project gaan onderzoeken hoe we Java classes zouden kunnen genereren die op een API teruggrijpen welke op een vergelijkbare manier werkt als de onderliggende API van Topbraid Composer OWL editor.

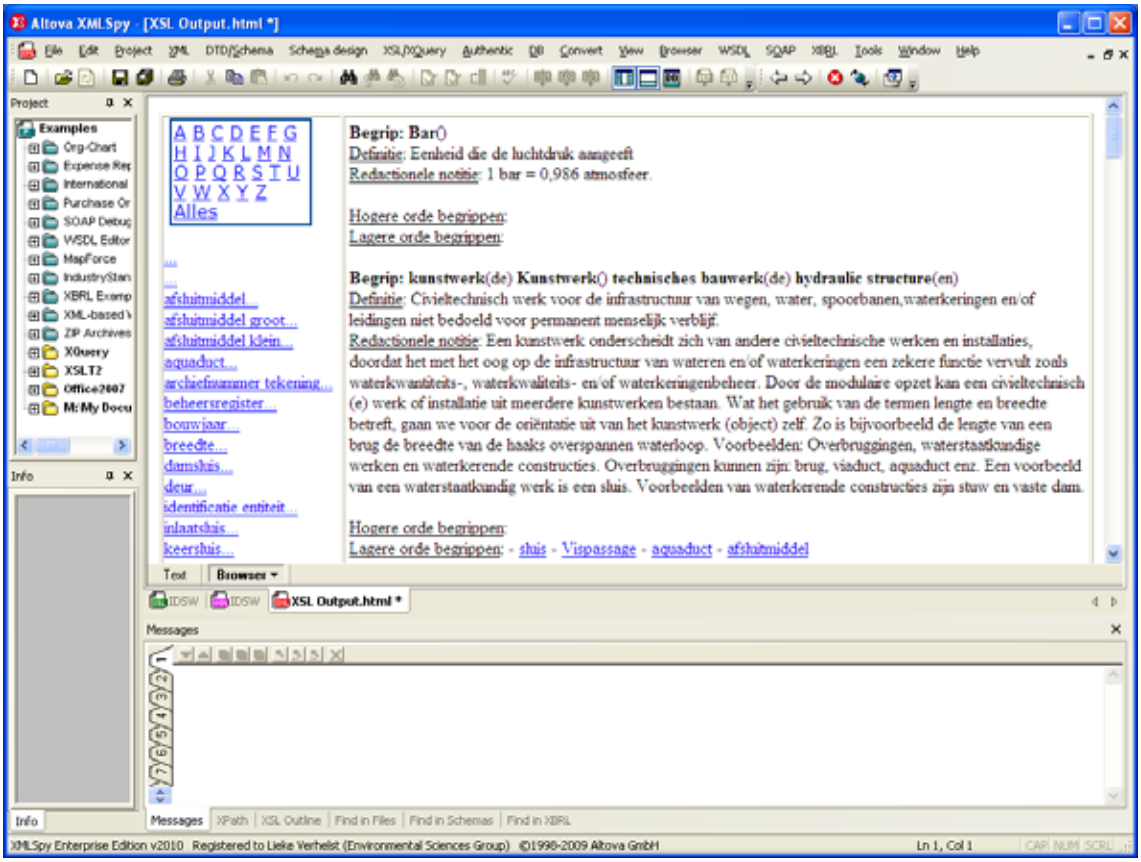
Ad 2.
Zelfs de meest uitgebreide versie van Topbraid Composer (Maestro editie) kan geen Java classes genereren zoals Protégé 3.4 dat doet. Omdat wij dit nogal vreemd vonden is contact gezocht met de leverancier om dit nader toe te lichten. De leverancier TopQuadrant [7]gaf aan dat zij een product suite kunnen leveren waarmee Semantische Webapplicaties kunnen worden gebouwd door het aanklikken van functionaliteit en zonder veel code te schrijven. Hiervoor moet wel hun eigen webserver worden gebruikt. Voor dit project is dit vanwege de kosten geen optie, maar we willen dit wel als mogelijkheid noemen.
Op het internet is een open source plugin beschikbaar [8] die wel deze functionaliteit biedt. Eenmaal geïnstalleerd in de Topbraid Composer omgeving, genereert deze OWL2Java plugin redelijk nette Java code die werkt op basis van de bijbehorende OWL2Java API. Ons bleek dat die API zonder problemen het OWL bestand inleest zoals dat is geschreven door Topbraid Composer. Vergeleken met de Protégé OWL API is deze OWL2Java API gemakkelijker werkend te krijgen en ook simpeler van opbouw. Vanwege het laatste kan het ontwikkeltraject van een applicatie relatief sneller worden doorlopen.

Ad 3.
XSLT is een transformatie protocol waarmee het ene XML formaat kan worden omgezet naar het andere XML formaat. Het kan ook worden toegepast om van XML HTML te genereren. Aangezien OWL een XML formaat is hebben we onderzocht of het publiceren van het OWL bestand via een XSLT vertaling naar HTML een werkbare

[7] http://www.topquadrant.com/
[8] http://www.incunabulum.de/projects/it/owl2java

methode is. Hiervoor is een XSLT document ontworpen dat HTML schrijft op basis van het OWL bestand. Omdat content dynamisch moet worden gepresenteerd moet de HTML pagina voorzien zijn van Javascript. Met deze methode zijn vergelijkbare resultaten te produceren als met de JSP methode. Wel moet rekening gehouden worden met de beperkingen van Javascript binnen een specifieke browser. Een voorbeeld daarvan is het gebruik van 'document.write'. Dit geeft problemen in Firefox, terwijl het wel werkt in Internet Explorer. Wat betreft de flexibiliteit: als de structuur van het OWL bestand wordt aangepast, moet de XSLT code ook worden aangepast. Dit houdt in dat in principe de data in de OWL mag veranderen zonder dat de XSLT code hoeft te worden aangepast.

Wat betreft performance ten opzichte van JSP komt deze methode negatiever uit. Aangezien het aan de client kant wordt geprocessed, zal het openen van een groot OWL bestand via XSLT lang duren om getoond te worden. Het resultaat van de conversie in Altova XML Spy is te zien in figuur 19. Omdat het om een proof of concept gaat zijn niet alle attributen opgehaald.



Figuur 19

XSLT transformatie van OWL getoond in Altova XMLSpy.

In een productieomgeving werkt dit als volgt: de OWL met de gekoppelde XSLT worden op een webserver geplaatst. Als via een webbrowser de OWL wordt aangeroepen wordt dit bestand meteen via XSLT vertaald naar een HTML pagina.

De gebruikte technieken zijn in de volgende relatieve vergelijkingstabel naast elkaar gezet:

Tabel 5
Relatieve vergelijking ontwikkelmethodes.

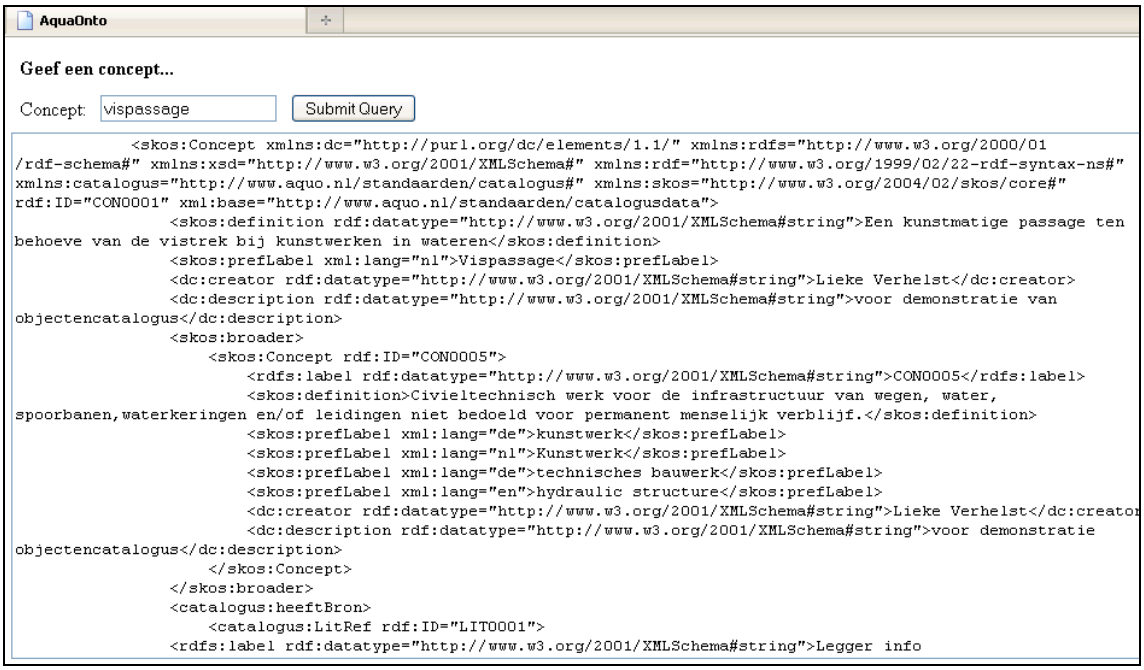
Techniek	Ontwikkeltijd (kort = +, lang = -)	Kosten (hoog = - , laag = +)	Flexibiliteit (flexibel = +, inflexibel = -)
Java gegenereerd door Protégé 3.4	-	++	-
OWL2Java	+/-	++	+/-
Topbraid Composer semantisch webserver product	+	- -	++
XSLT	+/-	++	+

4.6.1 De objectencatalogus OWL webservice

De geleverde webservice heeft als functie het in OWL formaat teruggeven van de eigenschappen en relaties van een concept wanneer een concept wordt gefeed aan de webservice.

De techniek die is gebruikt voor het maken van de webservice is gebaseerd op het queryen van XML. Hiervoor is XQuery gebruikt. Als ondersteunende engine wordt eXist-db gebruikt.[9] Exist is een Open Source database management systeem gebaseerd op XML technologie en is in hoge mate compliant is met de XQuery standaard.[10] Exist biedt meerdere mogelijkheden om webservices te maken met XQuery, o.a. met de XQueryServlet en de REST Server. Aanroepen van een XQuery script zorgt er voor dat de XQueryServlet deze leest en uitvoert. De REST servlet kan in de database op de server opgeslagen XQueries executeren. Voor de opgeleverde webservice is gebruikt gemaakt van de XQueryServlet van Exist. Van de Exist installatie inclusief het toegevoegde XQuery script en de OWL kan een distributie gebouwd worden (war bestand) welke eenvoudig onder bijvoorbeeld Apache Tomcat gedeployed kan worden. De oorspronkelijke Exist distributie maakt gebruik van Jetty als webcontainer. In het configuratiebestand web.xml worden bestanden met de extensie xql op de XQueryServlet gemapped.
Het XQuery script bouwt een HTML pagina op (figuur 20). Op de HTML pagina kan een concept (label) ingevoerd worden. Bij uitvoer haalt het script het gevraagde concept op en toont de OWL-XML in een textarea. Alleen de data en niet de definitie wordt geretourneerd. Vanuit de textarea kan de XML gekopieerd worden naar de omgeving waarin dit gewenst is.

[9] zie: http://exist-db.org/index.html
[10] http://www.w3.org/TR/xquery/


AquaOnto
Geef een concept...
Concept: vispassage
Submit Query
<skos:Concept xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdfs="http://www.w3.org/2000/01
/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:catalogus="http://www.aquo.nl/standaarden/catalogus#" xmlns:skos="http://www.w3.org/2004/02/skos/core#"
rdf:ID="CON0001" xml:base="http://www.aquo.nl/standaarden/catalogusdata">
<skos:definition rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Een kunstmatige passage ten
behoeve van de vistrek bij kunstwerken in wateren</skos:definition>
<skos:prefLabel xml:lang="nl">Vispassage</skos:prefLabel>
<dc:creator rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Lieke Verhelst</dc:creator>
<dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">voor demonstratie van
objectencatalogus</dc:description>
<skos:broader>
<skos:Concept rdf:ID="CON0005">
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">CON0005</rdfs:label>
<skos:definition>Civieltechnisch werk voor de infrastructuur van wegen, water,
spoorbanen,waterkeringen en/of leidingen niet bedoeld voor permanent menselijk verblijf.</skos:definition>
<skos:prefLabel xml:lang="de">kunstwerk</skos:prefLabel>
<skos:prefLabel xml:lang="nl">Kunstwerk</skos:prefLabel>
<skos:prefLabel xml:lang="de">technisches bauwerk</skos:prefLabel>
<skos:prefLabel xml:lang="en">hydraulic structure</skos:prefLabel>
<dc:creator rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Lieke Verhelst</dc:creator
<dc:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string">voor demonstratie
objectencatalogus</dc:description>
</skos:Concept>
</skos:broader>
<catalogus:heeftBron>
<catalogus:LitRef rdf:ID="LIT0001">
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Legger info

5 Conclusies en advies

5.1 Algemeen

Het semantisch model als proof of concept heeft inzichten gegeven waarmee de vooraf gestelde onderzoeksvragen kunnen worden beantwoord. De centrale vraag van het onderzoek was in hoeverre semantische datamodellen geschikt zijn voor de implementatie van het concept van de objectencatalogus. De proof of concept heeft aangetoond dat het door IDsW gespecificeerde model in OWL is vast te leggen. Ook is het mogelijk om de data uit de bestaande Aquo standaarden automatisch in te laden in het OWL design. Dat geldt voor alle bronnen. Voor de bronnen die niet direct een datastructuur hebben, zoals UM Aquo is de conversie wat complexer dan voor de bronnen met een duidelijke datastructuur.
Met de aldus verkregen proof of concept versie van de objectencatalogus is aangetoond dat een semantisch model contextinformatie toevoegt die in de huidige Aquo standaarden niet aanwezig is. Zie hiervoor met name paragraaf 4.3. Hierdoor is een gebruiker beter in staat de Aquo Standaarden te interpreteren en te gebruiken.
Per onderzoeksvraag kunnen we de volgende conclusies trekken:

– Zijn er bestaande en gepubliceerde formele ontologieën die een voorbeeld kunnen zijn voor de objectencatalogus en/of waarvan een deel kan worden gebruikt voor de objectencatalogus?

Echte formele ontologieën (domein ontologieën) zijn op dit moment niet gevonden. Wel op een meer generieke wijze is op dit moment SKOS, het Simple Knowledge Organizations Systems beschikbaar, een model dat bij uitstek geschikt is om gecontroleerde woordenlijsten semantisch te beschrijven. In SKOS zijn ook al thesauri geïmplementeerd, zoals GEMET (Milieu) en AGROVOC (Agrarisch). Deze kunnen vanwege de open SKOS structuur combinatie met de objectencatalogus gebruikt worden. De koppeling tussen SKOSsen kan worden gemaakt via de SKOS klasse ConceptScheme (zie paragraaf 4.2).

– Welke semantische frameworks, markup languages en datamodellen zijn geschikt voor het ontwerp van de objectencatalogus?

RDF is de basis markup language die in ons onderzoekresultaat is gebruikt. Hierbij werken we met relaties uit SKOS. Aanvullend is OWL Ontology Web Language nodig gebleken om de meer specifieke Aquo begrippen en relaties vast te kunnen leggen. Bovendien kan daarmee ook cardinaliteit en multipliciteit beschreven worden.

– Wat is het semantisch datamodel van de objectencatalogus (gebaseerd op het ontwerp op p. 18 van [1])?

Zie de uitwerking in paragraaf 4.2.

– Kunnen de eigenschappen, constraints en relaties zoals gedefinieerd in het ontwerp op p. 18 van [1] worden toegepast in het semantische model van de objectencatalogus ?

Zowel eigenschappen en relaties zijn in het semantisch model terug te vinden, voor de constraints moet nog een slag gemaakt worden.

– In hoeverre kan de data van de huidige Aquo standaarden worden geladen in het semantische model van de objectencatalogus?

Alle data kunnen geladen worden, waarbij de kanttekening geplaatst moet worden dat dit voor UM Aquo complexer is waardoor automatiseren lastig en tijdrovend is.

– Op welke wijze moet de data in het semantische model van de objectencatalogus worden beheerd?

Het beheer van het model kan het beste plaats vinden met de meer geavanceerde ontologie-editors, waarvoor met name Protégé als open source product beschikbaar is en TopBraid Composer als Marktproduct is aan te bevelen. Voor het beheer van de data zelf zijn deze editors door hun complexiteit minder geschikt en is een (web)client die afgestemd is op de objectencatalogus een beter alternatief. Voor het zoeken en browsen door de eindgebruiker (alleen viewen) is een webviewer (ook maatwerk) aan te bevelen.

– Als de objectencatalogus als invoerbronbestand wordt gebruikt, in hoeverre kunnen dan koppelingen worden gemaakt tussen bestaande Aquo standaarden en de objectencatalogus?

Door mapping van de huidige standaarden op de objectencatalogus zijn conversies mogelijk die vanuit één bron, de objectencatalogus, de huidige in gebruik zijnde standaarden moet kunnen voeden. Hiervoor moeten exportmodules worden ontwikkeld. Deze exportmodules zijn de inverse applicaties van degene die we nu voor het vullen van de catalogus gebruiken. Voor de conversie kunnen standaardtools worden gebruikt als FME, Altova Mission Kit en andere vergelijkbare marktproducten of door gebruik van XSLT in maatwerk-programmatuur.

– In hoeverre kan in het semantische model van de objectencatalogus de historie van de wijzigingen worden vastgelegd?

Historie is onderdeel van de structuur van de catalogus, waarmee voor de datahistorie vastligt. Historie heeft op het hoogste niveau een relatie met 'Thing' die op alle andere klassen overerft. Uit praktische oogpunten is het aan te bevelen om de historie van objecten niet in het semantisch model vast te leggen. Dit is nader beschreven in paragraaf 4.5.1.

Wat betreft het ontwikkelen van een webapplicatie om een OWL te ontsluiten kunnen we concluderen dat het maken van een applicatie op een OWL ontwerp (zonder data) afgestemd moet worden met de wijze waarop de OWL zelf is gemaakt. De structuur van de OWL wordt door verschillende ontologie-editors anders opgebouwd. Voor welke ontsluitingsmethode uiteindelijk wordt gekozen zal afhankelijk zijn van de weging van beslissings-factoren, zoals die in tabel 5 is weergegeven.
Wij adviseren verder om de data in de OWL niet in een productieomgeving als OWL bestand te serveren omdat het aantal triplets in de objectencatalogus te groot is voor een goede performance. Een achterliggende data-base is een goede oplossing hiervoor.

5.2 Aanbevelingen

Uit het onderzoek is gebleken dat in principe de semantische technieken waardevol en toepasbaar zijn. Maar dit is niet voldoende om aan te tonen dat de objectencatalogus nu al in de dagelijkse praktijk van IDsW kan worden ingevoerd. In het onderzoek zijn ook nieuwe issues, witte vlekken, boven gekomen die het naar onze mening wenselijk maken om enkele zaken nader te onderzoeken. Hierbij denken we in ieder geval aan de volgende onderwerpen:

1. het opnieuw tegen het licht houden van het ontwerp van de catalogus na de pilot;
2. gebruik van constraints;
3. de overgang van de oude situatie naar de nieuwe situatie;
4. het beheer van (de data in) de objectencatalogus.

5.2.1 Het opnieuw tegen het licht houden van het ontwerp van de catalogus

Voor het huidige ontwerp is het UML schema gebruikt dat door de IDsW is opgesteld. Het voordeel van het gebruiken van dit schema is dat dit goed de huidige samenhang van de Aquo-standaarden in beeld brengt. Toch verwachten we dat er een slag te maken valt waarmee de 'historische naamgeving' van begrippen kan worden verwijderd en de begrippen in de catalogus beter vindbaar zijn. Ook de ervaringen in de pilot kunnen het inzicht hierin vergroten. Voorbeelden zijn:

– het vereenvoudigen van de naamgeving van de in de objectencatalogus gebruikte klassen (bijvoorbeeld '(Vak)term' in plaats van Entiteit, etc.). Hierdoor gaat de oorspronkelijke relatie met de bron Aquo standaarden verloren, maar het is de vraag in hoeverre dit na enige tijd nog van belang is;
– de relatie Groep-Werkproces en Groep-Kennisgebied is nu een has-a relatie. Als er in de toekomst geen Groepen zijn te verwachten die zowel een Kennisgebied als Werkproces hebben dan is het 'Groep' als klasse overbodig en kan het beter een datatype property zijn van Werkproces respectievelijk Kennisgebied;
– de mogelijkheid van het gebruik van meerdere, gekoppelde SKOS concept schema's (voor LM Aquo, voor Aquo-lex, voor een deel van UM Aquo) en SKOS Collection voor de domeintabellen zou onderzocht moeten worden.

Dit zijn enkele voorbeelden die we nu kunnen benoemen. Denkbaar is dat deze lijst groter wordt met ervaringen die door het gebruik van de pilot objectencatalogus door IDsW worden opgedaan.

Verder is een nog openstaande vraag of toevoeging van inverse properties informatie toevoegt aan het model zoals dat in het proof of concept is ontworpen. Een voorbeeld is: heeftConcept en zijn inverse isConceptVan (de laatste is niet in het huidige model opgenomen). Als alle inverse relaties worden toegevoegd is het ontwerp lastiger te doorgronden vanwege de vele relatielijnen. In principe is de inverse property redundant als niet van belang is welke kant op wordt geredeneerd. Als het wel van belang is dat beide kanten op kan worden geredeneerd is toevoeging van de inverse relatie uiteraard noodzakelijk. Een toegevoegde waarde van een inverse relatie is verder dat door reasoning conclusies kunnen worden getrokken op basis van informatie die niet expliciet aanwezig is in het model. Stel dat wel in de data is vastgelegd dat 'de eenheid Bar heeft een Concept Bar' en dat niet in de data is vastgelegd 'het Concept Bar is concept van de eenheid Bar' en de inverse relatie 'isConceptVan' bestaat, dan kan door reasoning de tweede bewering worden afgeleid. Er ontstaat dan een nieuwe individual, de zogenaamde inferred (afgeleide) individual, die de tweede bewering in de data vastlegt. Onderzocht moet worden wat de toegevoegde waarde is van het afleiden van nieuwe informatie door middel van reasonen voor de praktijk van de objectencatalogus.

5.2.2 Gebruik van constraints

Het gebruik van constraints is in deze fase van het onderzoek niet meegenomen. De oorzaak hiervan ligt in de complexiteit van deze materie. Aan de ene kant kan het definiëren van een constraint in OWL lastig zijn, aan de andere kant is het handhaven van constraints in een OWL bestand geen eenduidige handeling: constraint checking gebeurt in de door ons onderzochte editors na het invullen van de data en de methode om dit te doen verschilt per editor (zie de paragrafen 3.3 en 4.5.1).

5.2.3 De overgang van de oude situatie naar de nieuwe situatie.

Met de kennis die is opgedaan over conversie strategieën van de in de Aquo standaarden gebruikte formaten naar OWL en andersom kan een migratiestrategie worden ontworpen. Hierbij wordt gespecificeerd hoe de bronbestanden op de meest efficiënte manier worden geïmporteerd in het OWL bestand. Verder kan worden bepaald welke data in de bronbestanden overbodig is geworden en in welke hoedanigheid de Aquo standaarden naast de objectencatalogus zullen blijven bestaan.
Er zal een duurzame koppeling kunnen worden ontworpen tussen de objectencatalogus als bronbestand en de overgebleven delen van de Aquo standaarden. Hierbij moet de integriteit van de data gegarandeerd worden.

5.2.4 Het beheer van (de data in) de objectencatalogus.

De voor en nadelen van het gebruik van een backenddatabase zouden moeten worden onderzocht, en de keuze voor het type database zou moeten worden bepaald. Ook zou er onderzoek gedaan moeten worden naar de voor- en nadelen van het ontwerpen van een alternatieve user interface voor data invoer en bewerking in plaats van de ontologie-editors. Onderzoek naar het gebruik van regels (constraints) in de ontologie is wenselijk, zodat onjuist gebruik van de objectencatalogus wordt voorkomen. Betrokkenheid van de domeinspecialisten van IDsW is hierbij noodzakelijk. Het resultaat van dit onderzoek naar constraints is een nieuwe versie van de objectencatalogus met constraints en testresultaten van reasoning op deze nieuwe objectencatalogus.

Wij stellen voor om een vervolgonderzoek te formuleren op basis van bovenstaande conclusies en aanbevelingen. Een vooraf gedefinieerde use case kan hierbij een leidraad zijn.

Literatuur

[1] 'Aquo catalogus' Voorstel, ontwerp, eisen, H-J. Lekkerkerk,IDsW, 02-08-2009

[2] W3C Semantic Web Activity, W3C, webpagina, http://www.w3.org/2001/sw/

[3] De family ontologie: http://www.owldl.com/ontologies/family.owl

[4] OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/

[5] SKOS Simple Knowledge Organization System Primer, W3C Working Group Note 18 August 2009, http://www.w3.org/TR/skos-primer/

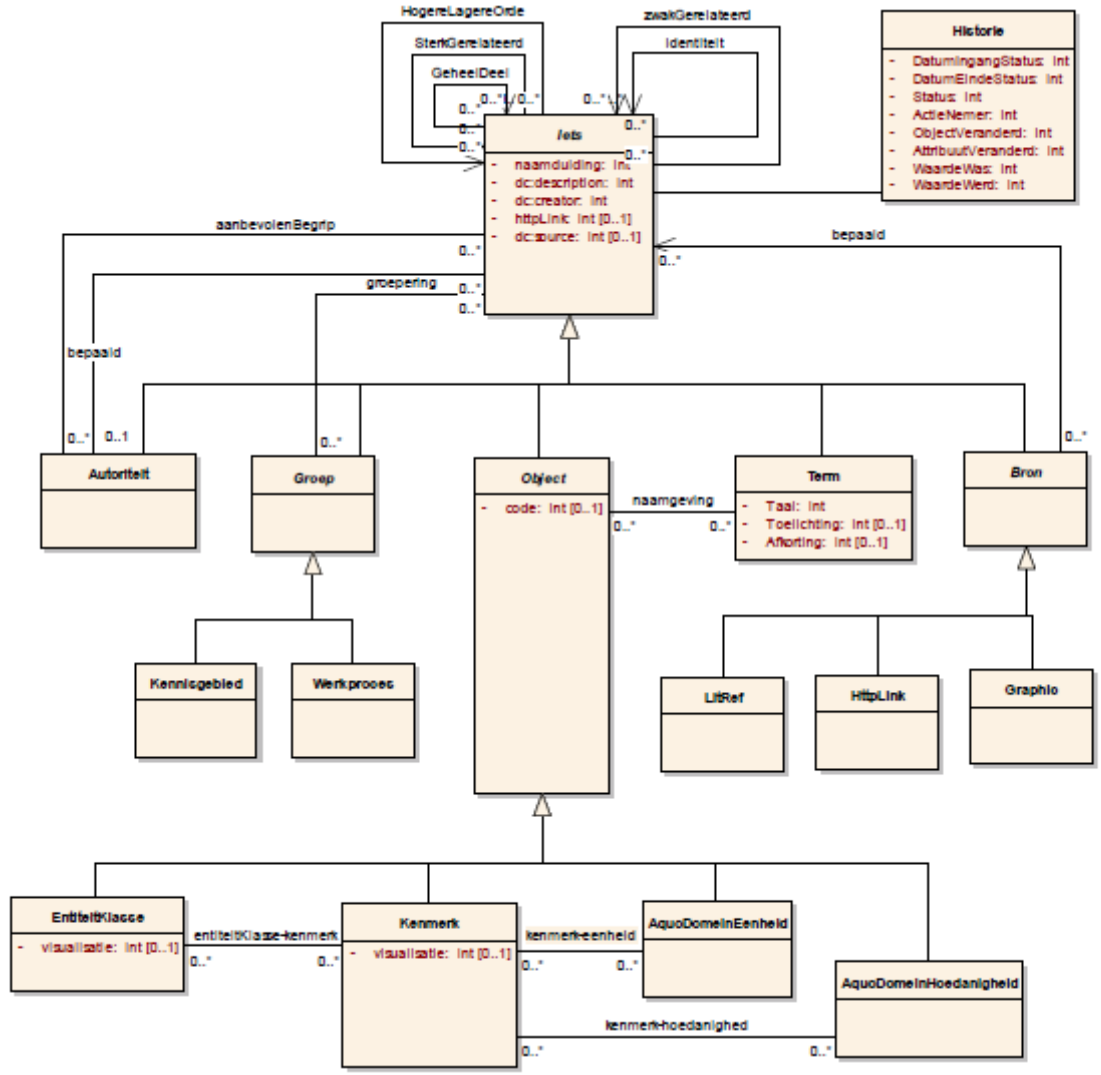
[6] Protégé, a free, open source ontology editor, http://protege.stanford.edu/

[7] Topbraid Composer, an enterprise-class modeling environment for developing Semantic Web ontologies and building semantic applications. http://www.topquadrant.com/products/TB_Composer.html

[8] DARPA Defense Advanced Research Projects Agency: http://www.darpa.mil/

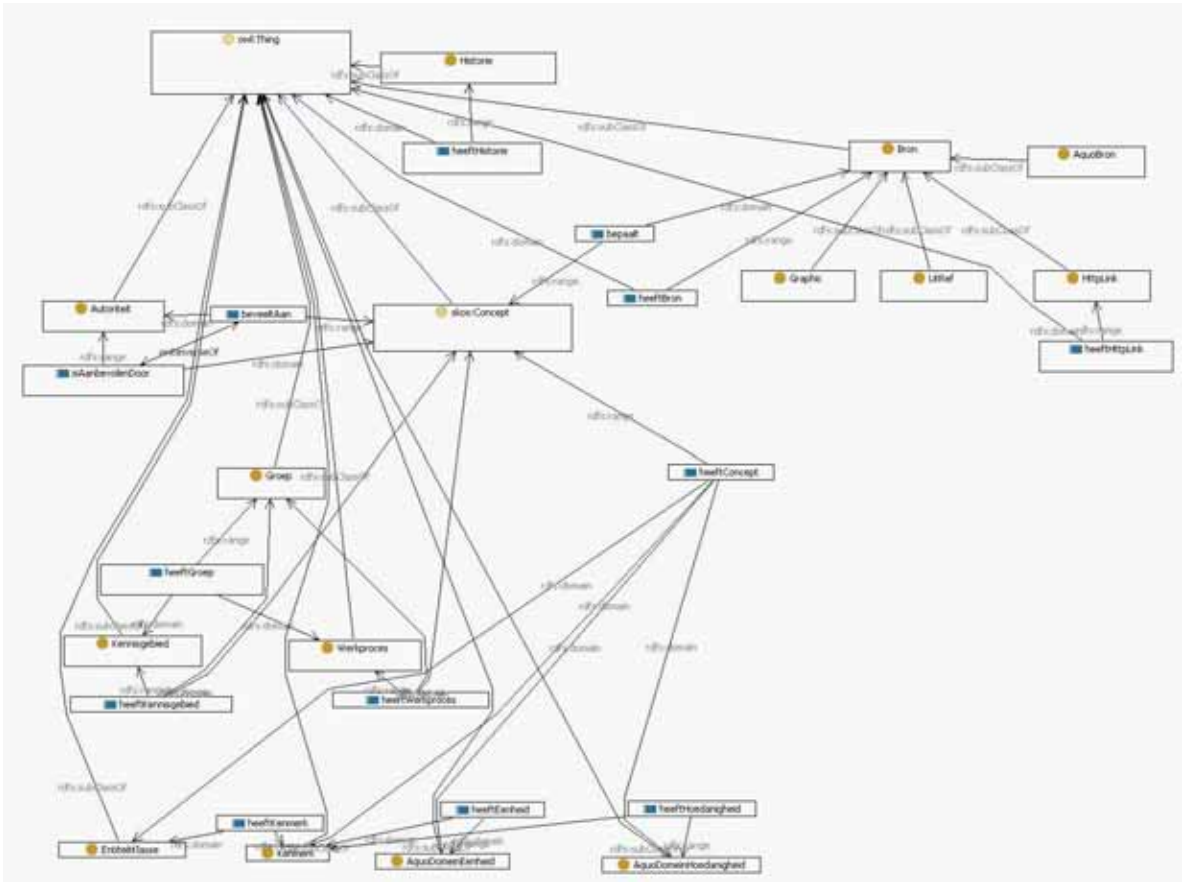
Bijlage 1 Ontwerpschema's

In deze bijlage zijn de ontwerpen van de objectencatalogus en het oorspronkelijke UML naast elkaar ter vergelijk weergegeven. De klassen en properties in de afbeelding van de objectencatalogus (figuur 22) zijn zo gerangschikt dat ze overeenkomen met de lokatie die ze ook in het UML hadden. Inhoudelijk is de figuur hetzelfde als figuur 7.



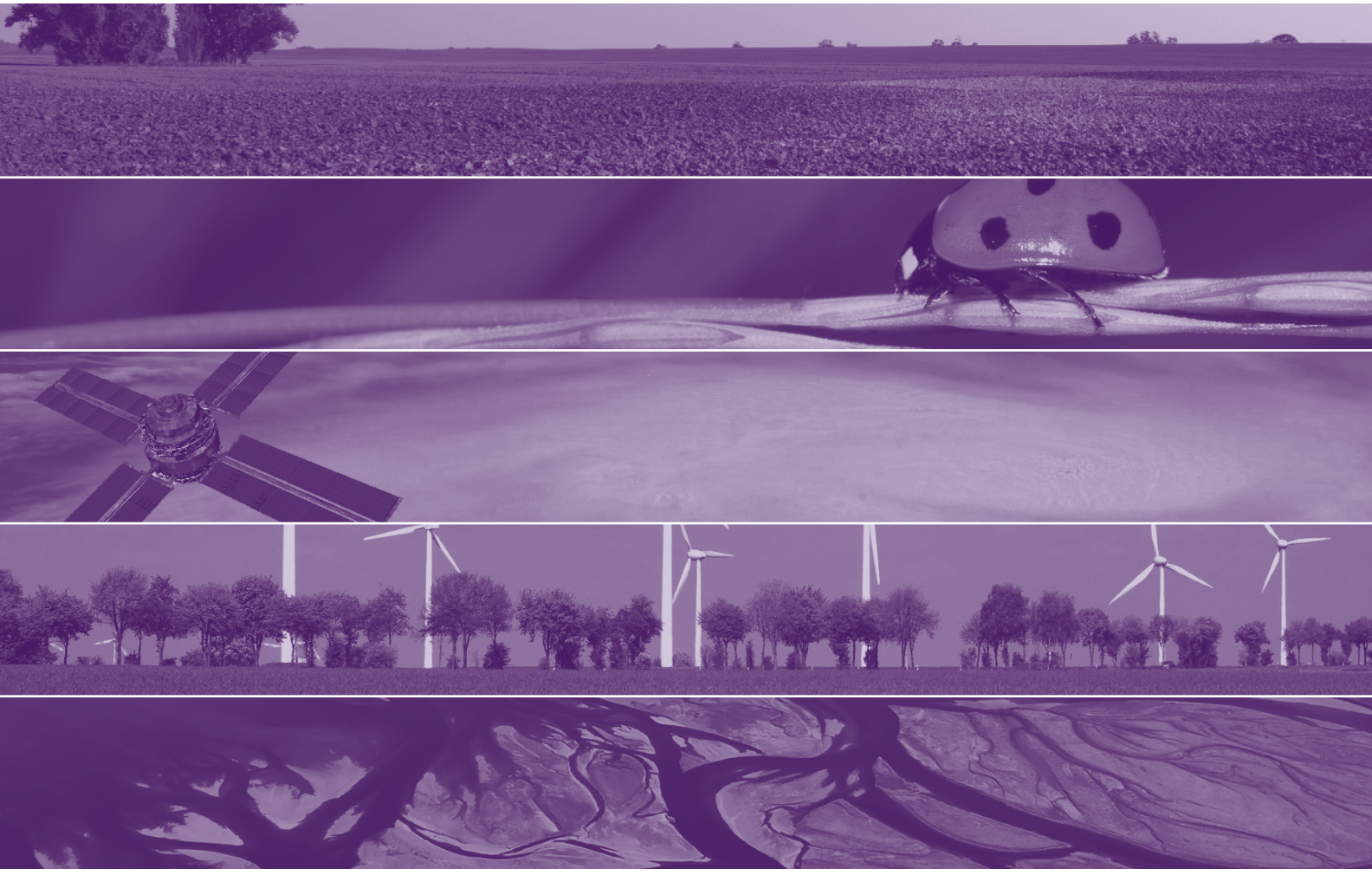
Figuur 21

Het oorspronkelijke UML ontwerp.



Figuur 22

Het OWL ontwerp van de objectencatalogus.



Alterra is onderdeel van de internationale kennisorganisatie Wageningen UR (University & Research centre). De missie is ‘To explore the potential of nature to improve the quality of life’. Binnen Wageningen UR bundelen negen gespecialiseerde en meer toegepaste onderzoeksinstituten, Wageningen University en hogeschool Van Hall Larenstein hun krachten om bij te dragen aan de oplossing van belangrijke vragen in het domein van gezonde voeding en leefomgeving. Met ongeveer 40 vestigingen (in Nederland, Brazilië en China), 6.500 medewerkers en 10.000 studenten behoort Wageningen UR wereldwijd tot de vooraanstaande kennisinstellingen binnen haar domein. De integrale benadering van de vraagstukken en de samenwerking tussen natuurwetenschappelijke, technologische en maatschappijwetenschappelijke disciplines vormen het hart van de Wageningen Aanpak.

Alterra Wageningen UR is hèt kennisinstituut voor de groene leefomgeving en bundelt een grote hoeveelheid expertise op het gebied van de groene ruimte en het duurzaam maatschappelijk gebruik ervan: kennis van water, natuur, bos, milieu, bodem, landschap, klimaat, landgebruik, recreatie etc.

Meer informatie: www.alterra.wur.nl