

NEN 3610 - Linked Data

Geonovum Standaard

Versie ter vaststelling 20 februari 2020



Deze versie:

<https://docs.geostandaarden.nl/nen3610/vv-st-nldp-20200220/>

Laatst gepubliceerde versie:

geen

Laatste werkversie:

<https://geonovum.github.io/NEN3610-Linkeddata/>

Redacteur:

Paul Janssen, [Geonovum](#)

Auteurs:

Linda van den Brink, [Geonovum](#)

Marco Brattlinga, [Ordina](#)

Marinus Vonhof, [Stichting RIONED en Sweco](#)

Niels Hoffmann, [Provincie Noord-Holland](#)

Pano Maria, [Skemu](#)

Hans Schevers, [Building Bits](#)

Ronald van Lanen, [Royal HaskoningDHV](#)

Joep van Genuchten, [Alliander](#)

Doe mee:

[GitHub Geonovum/NEN3610-Linkeddata](#)

[Dien een melding in](#)

[Revisiehistorie](#)

[Pull requests](#)

Rechtenbeleid:



Creative Commons Attribution 4.0 International Public License

(CC-BY)

Status van dit document

Deze paragraaf beschrijft de status van dit document ten tijde van publicatie. Het is mogelijk dat er actuelere versies van dit document bestaan. Een lijst van Geonovum publicaties en de laatste gepubliceerde versie van dit document zijn te vinden op <https://www.geonovum.nl/geo-standaarden/alle-standaarden>.

Dit is een definitief concept van de nieuwe versie van de standaard. Wijzigingen naar aanleiding van consultaties zijn doorgevoerd. De programmaraad van Geonovum beoordeelt dit definitief concept. Keurt zij het goed, dan is er een nieuwe standaard.

Inhoudsopgave

1. **Inleiding**
2. **Nederlandse LOD Cloud voor geo-datasets**
3. **Use case voor een NEN 3610 Linked Data profiel**
4. **Review beschikbare standaarden, handleidingen**
 - 4.1 ISO 19150-2
 - 4.2 INSPIRE RDF guidelines
 - 4.3 Spatial Data on the Web Best Practices
 - 4.4 Betekenisvol verbinden van informatie met BP4mc2-praktijkervaringen
 - 4.5 Linked Data Proxy (LDProxy)
 - 4.6 NTA8035
5. **Basisbeginselen van Linked Data**
 - 5.1 Introductie
 - 5.2 Identificatie
 - 5.3 Classificatie
 - 5.4 Normalisatie
 - 5.5 Verschillende type informatiemodellen
 - 5.6 Setoriëntatie in Linked Data en de relatie met de UML Class
 - 5.6.1 UML classes en OWL classes
 - 5.6.2 Wat is een Pand en wat is een Woonplaats
 - 5.6.3 Wat is een BAG-object?
 - 5.6.4 Als je met een OWL bril naar UML kijkt: wat is een Pand?
 - 5.6.5 Waar ontstaat het misverstand?
 - 5.7 Referentiedata
 - 5.8 Hergebruik van bestaande vocabulaires
6. **De NEN 3610 representatie in Linked Data**
 - 6.1 Verwijzingen
 - 6.2 Begrippen
 - 6.3 Klassen en eigenschappen
 - 6.4 RDF Gegevensregels
 - 6.4.1 URI template en NEN3610ID
 - 6.4.2 Temporele kenmerken en versies
 - 6.5 Gebruik van het NEN 3610 Linked Data model
 - 6.5.1 Maak gebruik van de NEN 3610 begrippen
 - 6.5.2 Maak gebruik van de NEN 3610 vocabulaire
 - 6.6 Metadata van datasets
7. **Transformatie van een NEN 3610-UML model naar Linked Data**
 - 7.1 Doel van de transformatie en methode
 - 7.1.1 Correct versus juist: handmatige vertaling is deels noodzakelijk
 - 7.1.2 Geautomatiseerde data-transformatie
 - 7.1.3 Geen roundtrip
 - 7.2 Bronmodel: NEN 3610 metamodel
 - 7.2.1 Toelichting op metamodel NEN 3610
 - 7.3 Transformatie: basisregels - encoding
 - 7.3.1 Inleiding
 - 7.3.2 Basis doelmodel: metamodel op basis van W3C standaarden

7.3.3	Basisregels voor transformatie
7.3.4	Standaard transformatieregels
7.3.4.1	Klassen
7.3.4.1.1	Klassen algemeen
7.3.4.1.2	Klasse met stereotype «featureType»
7.3.4.1.3	Klasse met stereotype «dataType»
7.3.4.1.4	Klasse met stereotype «union»
7.3.4.1.5	Klasse met stereotype «external»
7.3.4.1.6	Klasse met stereotype «enumeration»
7.3.4.1.7	Klasse met stereotype «CodeList»
7.3.4.2	Attributen
7.3.4.2.1	Attributen algemeen
7.3.4.2.2	Attribuut met stereotype «identificatie»
7.3.4.2.3	Attribuut met stereotype «voidable»
7.3.4.2.4	Attribuut met stereotype «materieleHistorie»
7.3.4.2.5	Attribuut met stereotype «formeleHistorie»
7.3.4.2.6	Attribuut met stereotype «materieleLevensduur»
7.3.4.2.7	Attribuut met stereotype «formeleLevensduur»
7.3.4.3	Associaties
7.3.4.3.1	Simpele associaties
7.3.4.3.2	Aggregatie en compositie
7.3.4.3.3	Generalisatie/Specialisatie
7.3.4.3.4	Associatieklassen
7.3.4.4	Kardinaliteit
7.3.4.5	Overig
7.3.4.5.1	Package
7.3.4.5.2	Constraint
7.4	Transformatie: specifieke aanpassing - encoding
7.4.1	Inleiding
7.4.2	Omgevingswet-DSO: specifieke transformatieregels
7.4.2.1	Metamodel DSO
7.4.2.2	Transformatieregels DSO
7.4.2.2.1	Klassen
7.4.2.2.1.1	Klassen algemeen
7.4.2.2.1.2	Klasse met stereotype «featureType»
7.4.2.2.1.3	Klasse met stereotype «dataType»
7.4.2.2.1.4	Klasse met stereotype «union»
7.4.2.2.1.5	Klasse met stereotype «external»
7.4.2.2.1.6	Klasse met stereotype «enumeration»
7.4.2.2.1.7	Klasse met stereotype «CodeList»
7.4.2.2.2	Attributen
7.4.2.2.2.1	Attributen algemeen
7.4.2.2.2.2	Attribuut met stereotype «identificatie»
7.4.2.2.2.3	Attribuut met stereotype «voidable»
7.4.2.2.2.4	Attribuut met stereotype «materieleHistorie»
7.4.2.2.2.5	Attribuut met stereotype «formeleHistorie»
7.4.2.2.2.6	Attribuut met stereotype «materieleLevensduur»
7.4.2.2.2.7	Attribuut met stereotype «formeleLevensduur»
7.4.2.2.3	Associaties
7.4.2.2.3.1	Simpele associaties
7.4.2.2.3.2	Aggregatie en compositie
7.4.2.2.3.3	Generalisatie/Specialisatie
7.4.2.2.3.4	Associatieklassen
7.4.2.2.4	Kardinaliteit

7.4.2.2.5	Overig
7.4.2.2.5.1	Package
7.4.2.2.5.2	Constraint
7.4.3	Bouwsector - Linked Data: Specifieke transformatieregels
7.4.3.1	Conceptueel Top Model NTA8035
7.4.3.2	Transformatieregels NTA 8035
7.4.3.2.1	Klassen
7.4.3.2.1.1	Klasse met stereotype «featureType»
7.4.3.2.1.2	Klasse met stereotype «dataType»
7.4.3.2.1.3	Klasse met stereotype «union»
7.4.3.2.1.4	Klasse met stereotype «external»
7.4.3.2.1.5	Klasse met stereotype «enumeration»
7.4.3.2.1.6	Klasse met stereotype «CodeList»
7.4.3.2.2	Attributen
7.4.3.2.2.1	Attribuut met stereotype «identificatie»
7.4.3.2.2.2	Attribuut met stereotype «voidable»
7.4.3.2.2.3	Attribuut met stereotype «materieleHistorie»
7.4.3.2.2.4	Attribuut met stereotype «formeleHistorie»
7.4.3.2.3	Associates
7.4.3.2.3.1	Simpele associaties
7.4.3.2.3.2	Aggregatie en compositie
7.4.3.2.3.3	Generalisatie/Specialisatie
7.4.3.2.3.4	Associatielassen
7.4.3.2.4	Kardinaliteit
7.4.3.3	Metamodel COINS
7.4.3.4	Transformatieregels COINS
7.4.3.4.1	Klassen
7.4.3.4.1.1	Klasse met stereotype «featureType»
7.4.3.4.1.2	Klasse met stereotype «dataType»
7.4.3.4.1.3	Klasse met stereotype «union»
7.4.3.4.1.4	Klasse met stereotype «external»
7.4.3.4.1.5	Klasse met stereotype «enumeration»
7.4.3.4.1.6	Klasse met stereotype «CodeList»
7.4.3.4.2	Attributen
7.4.3.4.3	Associates
7.4.4	Stedelijk Water - GWSW-OroX: Specifieke transformatieregels
7.4.4.1	Transformatieregels OroX: Maak een ontologie
7.4.4.2	Top-model GWSW-OroX
7.4.4.3	UML-concepten versus OroX-concepten
7.4.4.4	Klassen
7.4.4.4.1	Klassen algemeen
7.4.4.4.2	Klasse met stereotype «featureType»
7.4.4.4.3	Klasse met stereotype «dataType»
7.4.4.4.4	Klasse met stereotype «union»
7.4.4.4.5	Klasse met stereotype «external»
7.4.4.4.6	Klasse met stereotype «enumeration»
7.4.4.4.7	Klasse met stereotype «CodeList»
7.4.4.5	Attributen
7.4.4.5.1	Attributen algemeen
7.4.4.5.2	Attribuut met stereotype «identificatie»
7.4.4.5.3	Attribuut met stereotype «voidable»
7.4.4.5.4	Attribuut met stereotype «materieleHistorie»
7.4.4.5.5	Attribuut met stereotype «formeleHistorie»
7.4.4.5.6	Attribuut met stereotype «materieleLevensduur»

7.4.4.5.7	Attribuut met stereotype «formeleLevensduur»
7.4.4.6	Associaties
7.4.4.6.1	Simple associaties
7.4.4.6.2	Aggregatie en compositie
7.4.4.6.3	Generalisatie/Specialisatie
7.4.4.6.4	Associatieklassen
7.4.4.7	Kardinaliteit

8. Voorbeeld: UML-Golfbaan naar vocabulaire

- 8.1 Model van data Golfbaan conform: DSO
- 8.2 Model van data Golfbaan conform: OroX
- 8.3 Model van data Golfbaan conform: COINS

9. Aanpak voor het vergelijking van Linked Data datasets

- 9.1 Generatie opzet
- 9.2 Modelvergelijking
- 9.3 Linked Data modelstijlen
 - 9.3.1 Huidige stijlen in gebruik
 - 9.3.2 NEN 3610 use cases en requirements
- 9.4 Modelvergelijking op dataniveau
 - 9.4.1 Inleiding
 - 9.4.2 Gegenereerd COINS model
 - 9.4.3 Gegenereerd GWSW-OROX model
 - 9.4.4 Gegenereerd W3C model

A. Glossary**B. Referenties**

- B.1 Normatieve referenties
- B.2 Informatieve referenties

Samenvatting

Dit document is een combinatie van een technisch rapport, een handboek en een standaard. Vanwege het onderzoeks karakter van het onderwerp is er voor gekozen om deze drie typen documenten in één rapport, de standaard NEN 3610 - Linked Data, bij elkaar te houden. Het technische rapport beschrijft de samenhang en verschillen tussen de UML - Object-oriëntatie en Linked Data. In het handboek gedeelte worden de UML - RDF transformatieregels beschreven en in het normatieve gedeelte is de NEN 3610 - Ontologie opgenomen.

Een informatiestandaard en meer specifiek daarbinnen een informatiemodel, is een formele beschrijving van een toepassingsdomein binnen een digitale context. Op basis van informatiestandaarden wordt het proces van gegevensinwinning, gegevensdeling en het gebruik van gegevens ingericht. Informatiestandaarden vormen daarmee een belangrijk onderdeel van de architectuur van de digitale informatieoverheid, de i-overheid. In het raamwerk van geo-informatiestandaarden is het Basismodel Geo-informatie (NEN 3610) het handboek voor het opstellen van geo-informatiemodellen. NEN 3610 volgt hierin de aanpak van de ISO TC211 Geo-information/Geomatics waarin object oriëntatie als principe en UML modelleringstaal als richtlijn worden genomen om informatiemodellen in te beschrijven.

De afgelopen jaren zijn steeds meer use-cases ontstaan waarbij informatie uit verschillende domeinen aan elkaar moet worden gekoppeld. De i-overheid vraagt hier ook om. Informatiemodellen worden daarmee integraler van opzet. Hier ligt een uitdaging: tijdens de ontwikkeling van modellen voor specifieke informatiedomeinen worden impliciete aannames gemaakt die wel werken binnen een specifiek domein, maar data integratie tussen domeinen kan compliceren. Hiernaast speelt een tweede ontwikkeling een rol. Organisaties hebben steeds meer te maken met een veelheid van informatiestandaarden die elk vanuit eigen perspectief van belang zijn en ook behouden moeten blijven. In plaats van een veelheid van 1 op 1 transformaties of koppelingen tussen standaarden is men op zoek naar een mechanisme om met een algemeen koppelvlak betekenisvol ("semantisch") modellen aan elkaar te kunnen relateren.

Met (onder andere) deze uitdaging in gedachten, heeft het W3C een aantal 'recommendations' (standaarden) opgesteld omtrent Linked Data. Deze standaarden specificeren methoden om de aannames rond data definities meer expliciet te maken en daardoor op een machine-leesbare manier meer betekenis aan data te geven, specifiek met het oog op integratie van data uit verschillende bronnen. De implementatie-context is daarbij het web. Hiermee wordt dus een beweging van applicatie-gericht werken naar web-gericht werken ondersteund.

Er ontstaat hierbij een situatie dat binnen de wereld van informatiestandaarden de huidige manier van werken als "traditioneel" wordt bestempeld en Linked Data als de nieuwe, innovatieve oplossing wordt gepositioneerd. Dit document richt zich op de koppeling en integratie van beide methodes. Hier wordt inhoud aan gegeven door de NEN 3610 methodiek uit te drukken in Linked Data. Er is een NEN 3610 ontologie gemaakt en gepubliceerd die de basis legt voor een gezamenlijk vocabulaire voor geo-informatie-ontologieën. Deze ontologie is gepubliceerd als de standaard NEN 3610 ontologie. Daarnaast zijn er transformatieregels opgesteld waarmee de in NEN 3610 gedefinieerde en in UML uitgedrukte geo-informatiemodellen getransformeerd kunnen worden naar geo-informatie-ontologieën. Er wordt hierbij uitgegaan van internationale initiatieven die hier al voor ontwikkeld zijn o.a. bij INSPIRE, ISO en W3C. Door dit toe te passen en door te ontwikkelen in de Nederlandse context wordt deze kennis in Nederland geborgd in zowel de geo-uml als de geo-Linked Data community. Het levert ook weer kennis op die naar de internationale fora teruggestrakt wordt.

Het open samenwerkingsverband dat voor dit onderwerp door Geonovum is opgezet bestaat uit een verzameling van experts uit overheidspartijen die operationele toepassingen in Linked Data en geo-informatie ontwikkelen of beheren. Drie ontwikkelingen uit de praktijk zijn hierin richtinggevend: ervaringen bij Stedelijk Water (OroX), ervaring bij de bouw (NTA 8035 en COINS) en ervaringen bij de Omgevingswet (DSO). Deze soms ook verschillende methodes hebben een gezamenlijke basis in één generiek metamodel voor Linked geo-Data. Dit metamodel samen met het NEN 3610 metamodel is de basis voor een set aan generieke transformatieregels om een NEN 3610 informatiemodel te kunnen transformeren naar een eerste versie van een linked geo-data ontologie. Het resultaat hiervan is echter nog niet geoptimaliseerd voor linked data toepassingen en heeft maar beperkte operationele waarde. De data zijn als het ware nog maar beperkt gelinkt aan al bestaande vocabulaires en UML constructies werken onbedoeld en verkeerd door in de linked data ontologie. Door naast de generieke transformatieregels nog specifieke regels te ontwikkelen wordt de kwaliteit van de ontologie verhoogd en de

bruikbaarheid binnen de linked data context wel bereikt. De ontwikkelde transformatieregels zijn een handboek voor het omzetten van een NEN 3610-UML bronmodel naar een Linked Data doelmodel. De vertaling is naar de verschillende linked data vocabulaires die hiervoor van toepassing zijn: RDF, RDFS, SKOS, OWL en SHACL.

Door de transformatieregels te beschrijven en toe te passen op een fictief NEN 3610 geo-informatiemodel van een golfbaan, IMGolf, is er een realistische testcase waarin de semantiek van een NEN 3610-UML met de semantiek van een resulterende linked data ontologie wordt vergeleken. Door vervolgens ook een dataset van een golfbaan in beide modellen uit te drukken ontstaat een volledige test van werkelijkheid naar abstract model van de werkelijkheid naar een data representatie van de werkelijkheid.

1. Inleiding

Dit onderdeel is niet normatief.

Dit document is een combinatie van een technisch rapport, een handboek en een standaard. Vanwege het onderzoeks karakter van het onderwerp is er voor gekozen om deze drie typen documenten in één rapport, de standaard NEN 3610 - Linked Data, bij elkaar te houden. Het technische rapport beschrijft de samenhang en verschillen tussen de UML - Object-oriëntatie en Linked Data. In het handboek gedeelte worden de UML - RDF transformatieregels beschreven en in het normatieve gedeelte is de NEN 3610 - Ontologie opgenomen. Bij elk hoofdstuk is aangegeven of het normatief dan wel niet normatief is.

Steeds meer wordt Linked Data gebruikt als uitwissel- en publicatiemechanisme voor geo-informatie. NEN 3610 is de standaard voor het uitwisselen van geo-informatie, gebruikt Unified Modeling Language (UML) als formele taal voor het vastleggen van semantiek en beveelt Geography Markup Language (GML) aan als technisch uitwisselingsformat. NEN 3610 is hiermee nog niet geschikt om semantiek, gegevensdeling en uitwisseling middels Linked Data te realiseren.

NEN 3610 geeft ook aan dat indien een sector kiest voor een ander formaat dan GML en er nog geen codering van NEN 3610 naar dat technisch formaat bestaat, de sector gevraagd wordt deze codering te beschrijven.

In dit document wordt hier een begin mee gemaakt. Geonovum heeft het initiatief genomen om partijen die actief bij de toepassing van geo-informatie als Linked Data betrokken zijn samen te brengen rond het onderwerp NEN 3610 toegepast in Linked Data. Het doel daarvan is te komen tot een gezamenlijke werkwijze die zorgt voor interoperabiliteit tussen Linked geo-Data en een gecontroleerde relatie met het stelsel van NEN 3610 - informatiemodellen.

Het onderwerp is beperkt tot het ontwikkelen van de methode om een NEN 3610 model te transformeren, te implementeren in Linked Data. De relatie tussen het bronmodel, het NEN 3610 informatiemodel, en het doel, het begrippenkader en de ontologie, staat daarbij voorop. Het gaat dus niet zozeer om een algemene geo-Linked Data codering maar specifiek die van uit een NEN 3610 bronmodel.

Als resultaat zijn de volgende producten ontwikkeld:

1. De basisbeginselen van Linked Data en de relatie met UML - Object-oriëntatie (zie hfst. 5 [Basisbeginseln van Linked Data](#))
2. De NEN 3610 ontologie. De standaard NEN 3610 vertaald naar een LD implementatie. Het is hiemee de standaard manier om NEN 3610 in Linked Data toe te passen. (zie hfst. 6 [NEN 3610 representatie in Linked Data](#))

Hiermee wordt de ontologie gerealiseerd die als standaard gebruikt wordt om een NEN 3610 informatiemodel in Linked Data uit te drukken.

3. Transformatieregels of coderingsregels voor een vertaling van een NEN 3610-UML informatiemodel naar een Linked Data omgeving. (zie hfst. 7 [NEN 3610 representatie in Linked Data](#))

De transformatieregels zijn een handreiking voor het omzetten van een NEN 3610-UML bronmodel naar een Linked Data doelmodel. De vertaling is naar de verschillende Linked Data vocabulaires die hiervoor van toepassing zijn: RDF, RDFS, SKOS, OWL en SHACL.

Bij het maken van een informatiemodel staat de use case altijd centraal als kader waaraan een model moet voldoen. Dit geldt ook voor de transformatie van een NEN 3610 model naar Linked Data. De Linked Data toepassing kan een ander doel hebben dan met het NEN 3610 model bedoeld is. Er zijn wat dat betreft ook verschillende alternatieve oplossingen voor de transformatie mogelijk. Waar dat zo is geven we dat in dit document aan.

Het project is een open samenwerking tussen de geïnteresseerde partijen. Door middel van kennisdeling en aansluiting bij internationale ontwikkelingen wordt er toe gewerkt naar een nationaal en internationaal afgestemd resultaat.

Dit document beschrijft het NEN 3610 Linked Data profiel. Het betreft een standaard, als aanvulling op de bestaande NEN 3610 standaard. Het normatieve deel van deze standaard is beschreven in hoofdstuk zes. De overige hoofdstukken zijn informatief en bieden best practices in het gebruik van de standaard.

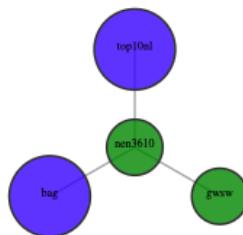
Leeswijzer: Het document begint met een beeld van hoe operationeel Linked Data ingezet wordt door middel van een inventarisatie van linked geo-datasets die er in Nederland beschikbaar zijn. Hoofdstuk drie schetst de gebruikstoepassing die met de toepassing van NEN 3610 en NEN 3610 modellen in een Linked Data omgeving bediend wordt en beschrijft de reden waarom een NEN 3610-LD profiel relevant is. Er zijn ook internationale initiatieven en standaarden die op het onderwerp codering van geo-informatie in RDF in gaan. Een kort overzicht daarvan en hoe die relateren aan dit onderwerp wordt in hoofdstuk vier gegeven.

De object-oriëntatie van NEN 3610 en de ontologische kennismodellen van Linked Data kennen verschillende paradigma's. De basisbeginselen van Linked Data en een referentie naar verdere documentatie geeft in hoofdstuk 5 een inleiding en overzicht over de impact van deze verschillen. Hoofdstuk 6 is normatief en bevat het NEN 3610-LD profiel en beschrijft het NEN 3610 vocabulaire in de relevante Linked Data vocabulaires. Dit profiel wordt vervolgens in hoofdstuk 7 toegepast en aangevuld met transformatieregels om een NEN 3610-UML informatiemodel te transformeren naar Linked Data. Hoofdstuk 7 is daarmee het inhoudelijke hoofdstuk dat de transformatieregels beschrijft. In dat hoofdstuk wordt ook geconstateerd dat er in Nederland drie verschillende Linked Data stijlen voor locatie informatie van toepassing zijn. Hoofdstuk 8 bevat een toepassing hiervan op het niveau van een NEN 3610 voorbeeldmodel IMGolf dat naar drie verschillende Linked Data modellen wordt getransformeerd. In het laatste hoofdstuk wordt aan de hand van het NEN 3610 voorbeeldmodel een dataset gecreeerd en na de transformatie toegepast en gepubliceerd als ontologie en begrippenkader conform de drie stijlen. Deze drie Linked Data stijlen kunnen hiermee worden vergeleken

2. Nederlandse LOD Cloud voor geo-datasets

Dit onderdeel is niet normatief.

Klik op onderstaand plaatje om een interactieve versie van de Nederlandse LOD Cloud te openen.



De intentie is om -als spin-off van de ontwikkeling van deze standaard- een Nederlandse LOD Cloud voor geodata te maken die ook gepubliceerd kan worden in de internationale [LOD Cloud](#).

De meest actuele versie van de Nederlandse LOD Cloud is [hier](#) te vinden. De bron voor deze LOD cloud is ook als Linked Data opvraagbaar: [lod-nen3610.ttl](#)

Voor het beschrijven van een dataset wordt gebruik gemaakt van [[geodcat-ap](#)], de Linked Data invulling van de ISO-19115 standaard voor het beschrijven van metadata over geometrische datasets.

3. Use case voor een NEN 3610 Linked Data profiel

Dit onderdeel is niet normatief.

De use case die we willen beantwoorden is:

Creëren van interoperabiliteit tussen NEN 3610 informatiemodellen opgesteld in UML enerzijds en de toepassing van dezelfde modellen in Linked Data.

Meer praktisch geformuleerd is dit te realiseren door:

Informatiemodellen en data publicatie conform NEN 3610 vertalen naar kennismodellen en data publicatie conform Linked Data.

Hiervoor biedt NEN3610LD twee producten:

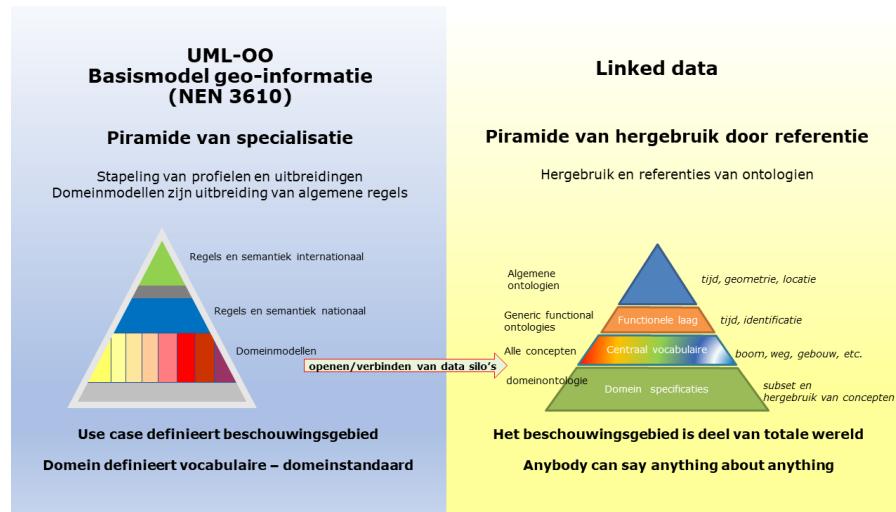
1. 1. Specificatie van het metamodel NEN 3610 als Linked Data ontologie: NEN3610-LD Ontologie.
2. 2. Een handreiking voor het transformeren van een UML-NEN 3610 informatiemodel naar Linked Data: NEN 3610-LD Transformatieregels.

Achtergrond:

Tot nu toe worden veel informatie standaarden in UML uitgedrukt. Het Object Oriëntatie paradigma heeft zich over de afgelopen decennia ontwikkeld als raamwerk voor het ontwikkelen van modellen van de werkelijkheid. Het model gericht werken, of model driven approach (MDA) zorgt ervoor dat vanuit het UML afgeleide producten waaronder implementatieschema's en software code gegenereerd kan worden. Voor het geo-informatiedomein heeft dit geleid tot standaardisatie in methodiek middels NEN 3610 en voor specifieke geo-informatie domeinen tot standaardisatie van semantiek. De use case is hierbij vooral gericht op informatie uitwisseling: data transport. In de implementatie daarvan staan applicaties centraal die data kunnen genereren, importeren, beheren en exporteren. In veel toepassingsdomeinen is deze standaardisatie gerealiseerd. De afgelopen jaren zijn steeds meer use-cases ontstaan waarbij informatie uit verschillende domeinen aan elkaar moet worden gekoppeld. Informatiemodellen worden daarmee integraler van opzet. Hier ligt een uitdaging: tijdens de ontwikkeling van modellen voor specifieke informatie domeinen worden impliciete aannames gemaakt die wel werken binnen een domein maar data integratie tussen domeinen kan complicerend. Deze aannames hebben invloed op de specifieke betekenis van de data: wat het wél vertegenwoordigt, maar vooral ook wat de data niet vertegenwoordigt. Deze aannames maken de datadefinities niet zozeer fout, maar het feit dat deze aannames impliciet zijn zorgt vaak voor problemen.

Met (onder andere) deze uitdaging in gedachten, heeft het W3C een aantal 'recommendations' (standaarden) opgesteld omrent Linked Data. Deze standaarden specificeren methoden om de aannames rond data definities meer explicet te maken en daardoor op een machine leesbare manier meer betekenis aan de data te geven, specifiek met het oog op integratie van data uit verschillende bronnen. De implementatie context is daarbij het web. Hiermee wordt dus een beweging van applicatie gericht werken naar web gericht werken ondersteund.

NEN 3610 is als standaard in het leven geroepen om geo-informatie eenduidig uit te kunnen wisselen tussen Nederlandse organisaties. Met het oog op bovenstaande ontwikkelingen wordt het relevant om te kijken hoe de NEN 3610 standaard in de context van Linked Data is uit te drukken om uiteindelijk het data integratie proces binnen en ook buiten het geo-domein te bevorderen. Van belang hierbij is ook te beseffen dat beide methoden elkaar niet uitsluiten maar juist complementair zijn in hun toepassing. UML-OO met informatiemodellen voor definitie van data uitwisseling in een gecontroleerde omgeving en Linked Data met kennismodellen als mechanisme voor data-integratie en publicatie in de open web-omgeving.



Figuur 1 Openen van de data silo's. 'UML-OO NEN 3610' en 'Linked Data methode' naast elkaar.

4. Review beschikbare standaarden, handleidingen

Dit onderdeel is niet normatief.

Er zijn een aantal standaarden en handleidingen die relevant zijn voor de implementatie van geo in Linked Data. Het gaat hierbij specifiek om standaarden die gericht zijn op de relatie tussen modellering van geo-informatie en implementatie daarvan in Linked Data. Van elke standaard wordt kort het toepassingsgebied, een samenvatting en de relevantie voor het NEN 3610 - LD profiel beschreven.

De volgende standaarden en handreikingen worden behandeld:

- ISO 19150-2: 2015 - Geographic information -- Ontology -- Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)
- INSPIRE - Guidelines for the RDF encoding of spatial data
- W3C - Spatial Data on the Web Best Practices
- Betekenisvol verbinden van informatie met BP4mc2-praktijkervaringen
- Linked Data Proxy (LDProxy)

4.1 ISO 19150-2

naam: ISO 19150-2: 2015 - Geographic information -- Ontology -- Part 2: Rules for developing ontologies in the Web Ontology Language (OWL)

url: <https://www.nen.nl/NEN-Shop/Norm/ISO-1915022015-en.htm>

type document: internationale standaard

organisatie: ISO TC/211

scope: Beschrijving van conversieregels van UML klassediagrammen (application schema's) naar OWL. Het betreft specifiek regels voor het vertalen van de UML klassediagrammen (in de meeste gevallen application schema's) die in de ISO 191xx set van geo-standaarden opgenomen zijn. Daarnaast zijn conversieregels beschreven voor conversie van informatiemodellen (application schema) gebaseerd op het General Feature Model van 19109.

omschrijving: Deel 2 in een set van ISO TC/211 Geo-informatie standaarden over gebruik van ontologieën als model en implementatie omgeving. De 19150-2 gaat specifiek over conversieregels van UML klassediagrammen naar OWL. Van alle UML constructies zijn conversieregels naar OWL opgenomen. De conversie wordt ondersteund door conversie-scripts, zie <https://github.com/ISO-TC211/GOM>. Alle in de 191xx standaarden opgenomen UML modellen zijn naar ontologieën vertaald en gepubliceerd op https://github.com/ISO-TC211/GOM/tree/master/isotc211_GOM_harmonizedOntology.

relevantie: Lijkt heel relevant omdat het onderwerp overeenkomt met dit NEN 3610-LD project: vertaalregels voor een NEN 3610 model naar Linked Data beschrijven. De 19150-2 neemt ook hetzelfde UML metamodel (GFM 19109) als NEN 3610 als bronmodel.

Er zijn echter weinig bekende praktische toepassingen van de 19150-2. Ook de ontwikkelde ontologieën worden niet echt gebruikt.

inhoudelijke opmerkingen: De 19150-2 geeft de conversieregels van UML naar OWL maar adresseert niet de use case, het gebruik van de gegenereerde ontologieën. Het blijft in de conversie heel dicht bij het oorspronkelijk UML en de ontologieën missen daardoor de aansluiting met de praktijk van Linked Data. Het levert als het ware correcte ontologieën op maar ze missen de Linked Data view daarop. Dit maakt het resultaat van de conversie weinig bruikbaar. Men zou verwachten dat bijvoorbeeld het spatial schema, 19107, het model van geometrietypen een bruikbaar RDF vocabulaire zou opleveren. Hier

wordt echter aan gewijfeld en men adviseert het gebruik van al bestaand RDF-vocabulaire op dit terrein, bijvoorbeeld GeoSPARQL.

Ondanks dit is het een goed startpunt voor het begrijpen van de conversie tussen UML en OWL. Het document Are3NA-RDF_vocabulary (https://ies-svn.jrc.ec.europa.eu/attachments/download/1188/ARE3NA-RDF_vocabulary_guidelines_Final.pdf) gaat verder in op de specifieke bruikbaarheid van de ISO 19150-2 maar neemt deze standaard ook als startpunt voor aanpassingen voor een beter toepasbare conversie.

4.2 INSPIRE RDF guidelines

naam: INSPIRE - Guidelines for the RDF encoding of spatial data

url: <http://inspire-eu-rdf.github.io/inspire-rdf-guidelines>

type document: richtlijn, best practice

organisatie: ARE3NA project "INSPIRE Re3ference Platform Phase 2"

scope: Beschrijving van UML-RDF coderingsregels voor representatie van INSPIRE datasets in RDF. Voor INSPIRE is RDF een mogelijk optionele implementatie omgeving die naast het aanbevolen GML een rol kan spelen. Dit onderzoek naar optionele RDF codering komt voort uit de ondersteuning van het algemene e-government programma en de open data community.

omschrijving: Het document beschrijft UML-RDF coderingsregels. De coderingsregels conformaten aan ISO 19118:2011 Geographic information - Encoding. Coderingsregels voor feature types, attributen en associaties zijn opgenomen. Over UML als input in de coderingsregels wordt gezegd dat het door INSPIRE gebruikte UML profiel (in principe ISO 19109) geen extensie nodig is om zinvol te kunnen coderen naar RDF. De output is een combinatie van RDF Schema en OWL. Voor de RDF serialisatie wordt Turtle aanbevolen. RDF/XML serialisatie is ook opgenomen. Van alle UML constructies, van package, feature type, attributen, associaties, complexe datatypen etc. worden de RDF/Turtle en RDF/XML serialisatie gegeven. Men gaat daarbij niet altijd uit van een standaard mapping maar merkt op dat het voor komt dat men naar de bedoeling van specifieke UML constructies moet kijken hoe die in RDF moeten komen. Bijvoorbeeld een datatype dat een versimpelde vorm van een feature type voorstelt moet in RDF een feature type worden. Men noemt dit de Permission REC/OWL/type/mapping/dataTypeAsSpatialObjectTypeRepresentation. Interessant is ook dat er een transformatietabel is van ISO 19107 geometrietypen naar GML ontologieklassen van GeoSPARQL (<http://www.opengis.net/ont/gml>). Bijvoorbeeld GM_Surface naar gmlowl:Surface.

Het onderzoeksrapport waarop deze guidelines zijn gebaseerd is [ARE3NA-RDF vocabulary guidelines](#).

relevantie: Zeer relevant voor dit onderzoek. Er worden expliciete uitgangspunten gedefinieerd die als basis gelden voor de UML - RDF coderingsregels. Er wordt een format gebruikt om de regels in te specificeren dat mogelijk hergebruikt kan worden. Het document is geschikt voor de automatische conversie maar heeft ook afwegingen om daar van af te wijken. Voor het beschrijven van de automatische transformatie van de NEN 3610 UML naar RDF wordt dit document als referentie gebruikt.

inhoudelijke opmerkingen: Alleen RDFS en OWL worden als vocabulaire gebruikt. Mogelijk is de codering vooral van uit het UML-GML perspectief ingestoken en mist het de RDF optimalisatie. Voor de NEN 3610 - UML RDF transformatie wordt dat toegevoegd.

4.3 Spatial Data on the Web Best Practices

naam: Spatial Data on the Web Best Practices [[sdw-bp](#)]

url: <https://www.w3.org/TR/sdw-bp/>

type document: best practice

organisatie: [W3C](#) en [OGC](#)

scope: geo-data / geo-informatie en web standaarden voor data. De scope is breder dan alleen Linked Data; het streven is in eerste instantie om geodata volgens de algemene webarchitectuur te publiceren. Dit kan RDF zijn, maar ook bijvoorbeeld (verrijkte) HTML.

omschrijving: Deze best practice geeft richtlijnen voor het publiceren van geodata op het web. De in totaal 14 best practices vertellen hoe je de algemene principes en architectuur van het Web moet toepassen op geodata; hoe je om gaat met specifieke geo-zaken zoals geometrie en coordinaatsystemen op het web; hoe je zorgt voor optimale toegang tot je data, en hoe je metadata voor geodata het beste kan publiceren. Alle best practices zijn gestoeld op de huidige praktijk.

relevantie: De relevantie voor geo-informatiemodellen is beperkt. De best practice gaat vooral in op het publiceren van data op het web en niet zozeer op informatiemodellen of hoe je [UML](#) modellen in OWL kunt uitdrukken. Er staan wel een aantal relevante aanbevelingen in het document:

- In [12.2.1](#) wordt verteld dat het belangrijk is om je geodata te publiceren met duidelijke semantiek; en in een Linked Data setting is de aanbeveling om eerst te zoeken naar bestaande [vocabulaires](#). Als het ontwikkelen van een eigen vocabulaire nodig is, link deze dan aan bestaande vocabulaires. Verder wordt verwezen naar de (algemene) [Data on the Web Best Practices](#).
- [Appendix A](#) geeft een overzicht van bestaande vocabulaires.

4.4 Betekenisvol verbinden van informatie met BP4mc2-praktijkervaringen

naam: Betekenisvol verbinden van informatie met BP4mc2-praktijkervaringen

url: <http://bp4mc2.org/>

type document: publicatie

organisatie: Platform implementatie Linked Open Data (PiLOD)

scope: best practices voor het toepassen van standaard Linked Data vocabulaires voor het beschrijven van een gegevenscatalogus. De scope is breder in de zin dat BP4mc2 ook betrekking heeft op datasets, datakwaliteit en gegevenscatalogi, waarbij NEN 3610 vooral focust op gegevensmodellen. BP4mc2 gaat niet specifiek in op geometrische vraagstukken.

omschrijving: Publicatie over het proces om data op het web te presenteren. Linked Data is daarbij de methode. Inzicht wordt gegeven in de relatie tussen betekenis van gebeurtenissen in de werkelijkheid en hoe die vertaald worden in begrippen en semantiek middels de grammatica van [SKOS](#) en [OWL](#). De uri als sleutel voor objecten in een informatiesysteem en de rol van registers worden benadrukt. De publicatie geeft een uiteenzetting van theorie en praktijk over Linked Data, de methode, begrippen en manier van denken en werkwijze. Er is een verschil en een relatie tussen een datamodel (gegevens) en een model (begrippen) van de werkelijkheid. UML wordt daarbij gepositioneerd als methode voor het datamodel en SKOS-OWL voor het begrippenmodel. De architectuur van een Linked Data implementatie wordt toegelicht.

relevantie: De achtergrond en filosofie van Linked Data wordt beschreven in de context van een informatievoorziening middels web-standaarden. Begrippen worden uitgelegd en de relatie tussen data, begrippen en de werkelijkheid wordt beschouwd. Een model van data is iets anders dan een model van begrippen. UML en SKOS-OWL worden daarbij aan elkaar gerelateerd.

4.5 Linked Data Proxy (LDProxy)

naam: Linked Data Proxy (LDProxy)

url: <https://hub.docker.com/r/iide/ldproxy/>

type document: software

organisatie: Is ontwikkeld binnen het programma European Location Interoperability Solutions for e-Government (ELISE)

scope: datauitwisseling met adapter voor WFS

omschrijving: LDProxy is een adapter op een WFS service die zorgt voor een RESTful API die naast de WFS-GML additionele output formats als GeoJson, HTML and JSON-LD realiseert. Deze dataformats worden on the fly gecreëerd op basis van de WFS data. De LDProxy is ontwikkeld om de bestaande WFS services te verbeteren op basis van Spatial Data on the Web Best Practices. Parallel daaraan is ook de draft WFS 3.0 specificatie ontwikkeld die voor een deel ook in de LDProxy is verwerkt.

relevantie: Deze adapter laat het resultaat zien van de LD publicatie van geodata die middels on the fly mapping uit GML gegenereerd worden. Het resultaat kan met het GML origineel en de formele UML modellen in de dataspecificaties vergeleken worden. Dit helpt om een notie te krijgen van de verschillende formats en hun relatie. In de 'on the fly mapping' wordt er waarschijnlijk gebruik gemaakt van de transformatieregels uit het document INSPIRE - Guidelines for the RDF encoding of spatial data. Het zal dan alleen gaan om de automatische transformatieregels en niet afwijkende regels die van uit specifieke domein UML/XSD nodig zijn. Het is interessant om het resultaat van de LDProxy mapping te vergelijken met de mapping die in dit document is uitgewerkt.

4.6 NTA8035

naam: NTA8035 - Semantische Gegevensmodellering en Integratie in de Gebouwde Omgeving

url:

type document: Nederlandse Technische Afspraak

organisatie: NEN NC 381184 werkgroep

scope: Afsprakenstelsel (Conceptueel Meta Model en Conceptueel Top Model) vanuit het perspectief van partijen uit de Gebouwde Omgeving in Nederland.

omschrijving: Het doel van deze Nederlandse Technische Afspraak (NTA) is om afspraken vast te leggen voor de toekomstvaste semantische modellering en integratie (uitwisseling of deling) van digitale gegevens (Engels: "data") op basis van W3C "Linked Data/Semantic Web (LD/SW)" standaarden.

Het gaat hierbij niet alleen om standaard gegevensformaten en benaderingsmethoden maar vooral ook om gegevensstructuren (Engels: "data models") die betekenis ("semantiek") geven aan de gegevens.

Het zijn afspraken opgesteld vanuit het perspectief van partijen uit de "gebouwde omgeving" (gebouwen en infrastructuren) in Nederland en waar partijen op voort kunnen borduren m.b.t. hun eigen ontwikkelingen met waarborging op interoperabiliteit tussen deze partijen bij toepassing.

relevantie: De modellen die conform NTA8035 gemodelleerd worden raken ook de modellen die in onder het NEN3610 stelsel vallen, zoals IMGeo en IMBor. Het is dus van belang dat het NEN3610-LD profiel interoperabel is met de NTA8035.

5. Basisbeginselen van Linked Data

Dit onderdeel is niet normatief.

5.1 Introductie

We veronderstellen enige kennis van wat Linked Data nu eigenlijk is en wat we bijvoorbeeld bedoelen met de term triples.

Als eerste introductie van Linked Data zijn dit goede bronnen:

- [Beknopte uitleg op Nora Online](#)
- [Uitleg in video](#) (Door Manu Sporny, Engels)
- [Uitleg om te lezen](#) (Door Lieke Verhelst, Nederlands)

NOOT

In dit document gebruiken we de term vocabulaire wanneer we het in algemene zin hebben over een verzameling van termen waarmee je je uitdrukt. De term begrippenkader gebruiken we voor een model van begrippen die (tekstueel) gedefinieerd zijn en onderlinge relaties hebben, zoals bijvoorbeeld in een thesaurus. Met de term ontologie bedoelen we een machine-leesbaar kennismodel met een set regels, die gebruikt kunnen worden om extra kennis af te leiden uit gelinkte data. Zowel vocabulaires als ontologieën zijn in ons taalgebruik dus 'vocabulaires'.

Voor het begrijpen van de rest van dit document is het belangrijk om te weten dat het gedachtegoed van Linked Data fundamenteel verschilt van het gedachtegoed van de administratieve systemen die we kennen zoals in relationele databases en object-oriëntatie. In Linked Data doen we beweringen over specifieke dingen uit de werkelijkheid of, om precies te zijn, uit het beschouwingsgebied ofwel 'universe of discourse'. De dingen waar we beweringen over doen hebben een identificerende URI (meer hierover in § 5.2 Identificatie). Elke 'triple' is een bewering die invulling geeft aan één specifieke eigenschap van een onderwerp, het subject.

Naast een subject bestaat elke bewering of triple uit een eigenschap en een waarde. De waarde kan ook weer een onderwerp op zich zijn met een eigen unieke sleutel (URI) en eigenschappen. Het bijzondere van Linked Data is dat dat andere onderwerp heel ergens anders opgeslagen kan zijn: het hoeft niet in dezelfde database te zitten. Dit betekent ook dat iedereen beweringen kan doen over elkaar's onderwerpen: [Anyone can say anything about anything \(AAA\)](#). Hieruit volgt weer dat je niet 'alles' kunt weten over een onderwerp (er kunnen immers beweringen bestaan die je niet gevonden hebt): dit heet het open wereld principe.

Data modelleren in OWL lijkt op het eerste gezicht best veel op UML. Je hebt klassen, eigenschappen, en relaties. Er is echter een fundamenteel verschil. Met OWL beschrijf je de regels van hoe de wereld in elkaar zit en ga je vervolgens met die regels data achteraf classificeren. Je bent dus eigenlijk geen data aan het modelleren, maar kennis aan het beschrijven die kan worden toegepast op data. In de object-georiënteerde / relationele wereld kan data niet bestaan zonder expliciet een instantie te zijn van een klasse en is je data van tevoren gecategoriseerd. Als je in een UML model bijvoorbeeld de klasse fiets modelleert als een object met precies twee wielen, heeft dit als resultaat dat een instantie van een fiets met één of drie wielen wordt afgekeurd: deze voldoet niet aan de definitie van fiets. Over andere objecten met wielen zegt dit niets, tenzij het UML model daar ook klassen voor definieert. In Linked Data zou het resultaat echter zijn dat elk ding dat twee wielen heeft, behoort tot de klasse fiets. Dus ook een motorfiets, een step, of een tweewielige kruiwagen. Een ding met één of drie wielen is geen 'incorrecte instantie' maar blijkbaar een ding dat tot een andere, onbekende klasse behoort.

Een informatiemodel in de Linked Data wereld kan bestaan uit verschillende onderdelen:

- Een model van begrippen, dat beschrijft welke concepten er in het domein dat je wilt modelleren een rol spelen, en wat ze betekenen. Dit kun je uitdrukken met SKOS [skos-primer].

- Een *ontologie*, die een kennisdomein beschrijft in termen van klassen en eigenschappen die relevant zijn binnen dit kennisdomein, aangevuld met *regels*, die gebruikt kunnen worden om extra kennis af te leiden uit gelinkte data. Met behulp van een ontologie kunnen computers begrijpen wat de data betekent en redeneren over data. Een ontologie kun je uitdrukken met [RDFS \[rdf-schema\]](#) en [OWL \[owl2-primer\]](#).
- Een specificatie van de *structuur* die de gegevens in een Linked Data dataset hebben, bijvoorbeeld eigenschappen die altijd aanwezig zijn voor een specifieke klasse, [datatypen](#), cardinaliteiten, enzovoort. Dit kun je uitdrukken met [SHACL \[shacl\]](#).

Het bijzonder aan Linked Data is dat het niet nodig is om een uitwisselingsmodel te beschrijven. Alle Linked Data, ongeacht het model, kan uitgewisseld worden met een aantal gestandaardiseerde uitwisselingsformaten. De bekendste formaten zijn Turtle, RDF/XML en JSON-LD

De hierboven beschreven uitgangspunten van Linked Data hebben verregaande consequenties die informatiemodelleren voor Linked Data fundamenteel anders maken dan informatiemodelleren voor een gesloten administratief ecosysteem. Wil je dat jouw Linked Data bruikbaar is voor anderen, dan zijn er verschillende aspecten waar je rekening mee moet houden.

5.2 Identificatie

In Linked Data gebruiken we [IRIs](#) om dingen te identificeren. IRIs zijn een goed mechanisme om wereldwijd unieke sleutels aan [subjecten](#) toe te kennen en maken het mogelijk om links te leggen. Het is belangrijk om goed na te denken over de opbouw van IRIs als je deze aan je data gaat toekennen. Ze moeten in ieder geval uniek en persistent zijn; bovendien is het gewenst dat ze *derefereable* zijn, zodat je op het desbetreffende web adres bruikbare informatie over het onderwerp kunt vinden. Meer lezen hierover kun je in onderstaande bronnen:

- [URI strategie NL](#) (Nederlands)
- [Best practices over Spatial Data Identifiers](#) (Engels)
- [Good URIs for Linked Data](#) (Engels)
- [Cool URIs for the Semantic Web](#) (Engels)

NOOT

Een IRI is gewoon een URI, maar beter internationaal bruikbaar omdat je in een IRI ook tekens uit niet-latijns schrift kan gebruiken. En een URI is soms ook gewoon een URL, dwz: als de identificatie ook een locatie op het web is (meestal begint zo'n URI dan met [http](http://), zoals in de adresbalk van je browser).

Het is daarnaast ook belangrijk om precies te zijn over het [subject](#) dat een IRI identificeert. Een verschil met administratieve systemen is dat het subject van een Linked Data bewering verwijst naar een ding uit het [beschouwingsgebied](#). Het is in Linked Data belangrijk om heel precies te zijn over wat het onderwerp, het 'subject' van een [triple](#), precies is. Een triple over een persoon heeft een ander onderwerp dan een triple over een document dat gaat over die persoon. In administratieve systemen zijn we niet gewend om zo precies te zijn: we impliceren dat een bestand met persoonsgegevens over een specifieke persoon gaat, zonder dat we direct de persoon identificeren. Vaak wordt in Linked Data onderscheid gemaakt tussen [information resources](#) en [non-information resources](#). Een document of een database record over een persoon (de information resource) is iets anders dan de persoon zelf (de non-information resource). Als het bijvoorbeeld belangrijk is om onderscheid te maken tussen formele en materiële historie, zou je dit in Linked Data doen door deze twee vormen van historie aan twee verschillende subjecten te hangen: de geboortedatum aan de persoon, en de ontstaansdatum van de informatie aan het record of document.

Er is nog een ander belangrijk verschil tussen UML en OWL. Dit heeft te maken met de Unique Naming Assumption ([UNA](#)) in UML. Daarmee wordt namelijk gesteld dat een unieke ID naar een unieke entiteit verwijst en dat een unieke entiteit door één unieke ID wordt aangeduid. Dit betekent dat 2 objecten met 2 verschillende ID's nooit naar hetzelfde kunnen, dan wel mogen, verwijzen. In Linked Data is dit laatste wél mogelijk. Het idee is namelijk dat meerdere mensen iets over hetzelfde ding kunnen zeggen, en dan kun je er vanuit gaan dat ze daar ook een andere ID voor hebben gebruikt. Door de [owl:sameAs](#) relatie kan in Linked Data aangegeven worden dat twee verschillende ID's naar hetzelfde ding verwijzen.

NOOT

Consequenties voor NEN 3610

In NEN 3610:2011 zijn regels opgenomen voor identificatie. Deze regels moeten worden aangepast voor het toepassen van IRIs als identificatie, aangevuld met een regel en/of aanbevelingen voor het toepassen van meerdere unieke IDs voor hetzelfde ding in de werkelijkheid. Ook is het gewenst om het belang uit te leggen van het precies modelleren van het subject, zonder daarbij bijvoorbeeld het ding in de werkelijkheid en het object in de registratie samen te voegen in één klasse.

5.3 Classificatie

Het opstellen van een ontologie wordt ook wel "ontology engineering" genoemd. Bij het opstellen van een ontologie maak je expliciet welke kennis uit een domein relevant is. Een basisbeginsel daarbij is dat daadwerkelijk het domein wordt beschreven, en niet zozeer de gegevens die daarover opgeslagen worden. Een belangrijk instrument is daarbij de *classificatie*. Classificatie gaat over het benoemen van groepen (de "klassen") van dingen die een aantal eigenschappen gemeen hebben. Zo hebben "dingen" uit de groep van "golfbanen" gemeen dat deze uit "holes" bestaat. Dus blijkbaar is de relatie van belang tussen een "golfbaan" en een "hole". Daarnaast geldt voor de groep van "bosbanen" dat er veel bomen op het golfbaanterrein staan. Dus blijkbaar is een eigenschap als "bomen op de golfbaan" van belang. Tenslotte weten we dat elke bosbaan ook een golfbaan is, en daarmee kunnen we stellen dat de klasse van bosbanen een subklasse is van de klasse van golfbanen. Voor elke klasse moet gelden dat er een eigenschap is waarmee we elk exemplaar van deze klasse uniek kunnen onderscheiden van de exemplaren uit een andere klasse. In het geval van de klasse van "bosbanen" kunnen we dat op basis van de eigenschap "bomen op de golfbaan". Het opstellen van een ontologie omvat dus het beschrijven van de groepen die we relevant vinden om te onderscheiden, de eigenschappen en relaties die daarbij een rol spelen, en regels waarmee je bepaalt hoe je op basis van de eigenschappen en relaties kunt bepalen tot welke groep iets behoort.

Klassen, eigenschappen en relaties uit een ontologie lijken verdacht veel op klassen, attributen en associaties uit UML. Bij UML 'construeren', maak, je echter klassen als een blauwdruk voor de instanties die je gaat uitwisselen. Het verschil zit er in dat je in een ontologie over het algemeen verder gaat met het opdelen in klassen, en ook eigenschappen benoemd die wel nodig zijn om je domein te "begrijpen", maar niet per se nodig zijn om informatie over het domein te delen. Zo is het voor ons golfbanen voorbeeld voldoende om vast te leggen tot welk "type" een golfbaan behoort (zoals: bosbaan, polderbaan, etc), maar niet waarom dit nu zo is. Andersom maak je in UML soms modelleerde keuzes die specifiek betrekking hebben op de manier waarop je de gegevens wilt vastleggen.

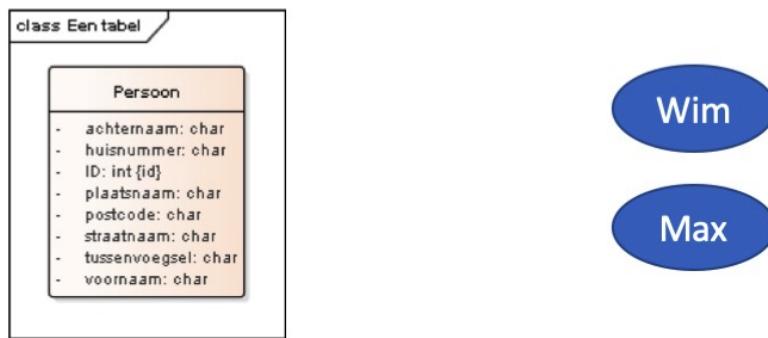
5.4 Normalisatie

Zoals uitgelegd in de vorige paragraaf, is het in Linked Data belangrijk om in onze modellering van de klassen waartoe de subjecten uit triples kunnen behoren, heel precies te zijn en zo dicht mogelijk bij de werkelijkheid te blijven. Voordat je begint met het transformeren van een informatiemodel naar een Linked Data model is het daarom belangrijk om stil te staan bij het oorspronkelijke doel en de uitgedrukte betekenis van het informatiemodel. Is het een conceptueel of logisch datamodel? Een technisch objectmodel? Een technisch berichtmodel? De meeste informatiemodellen zijn in meer of mindere mate gedenormaliseerd. Dit is omdat de informatiemodellen vanuit een bepaald oogpunt zijn opgesteld en vaak een technisch insteek hebben. Bijvoorbeeld een berichtmodel voor geoptimaliseerde communicatie, waarbij selectief aangebrachte redundantie voordeelig kan zijn.

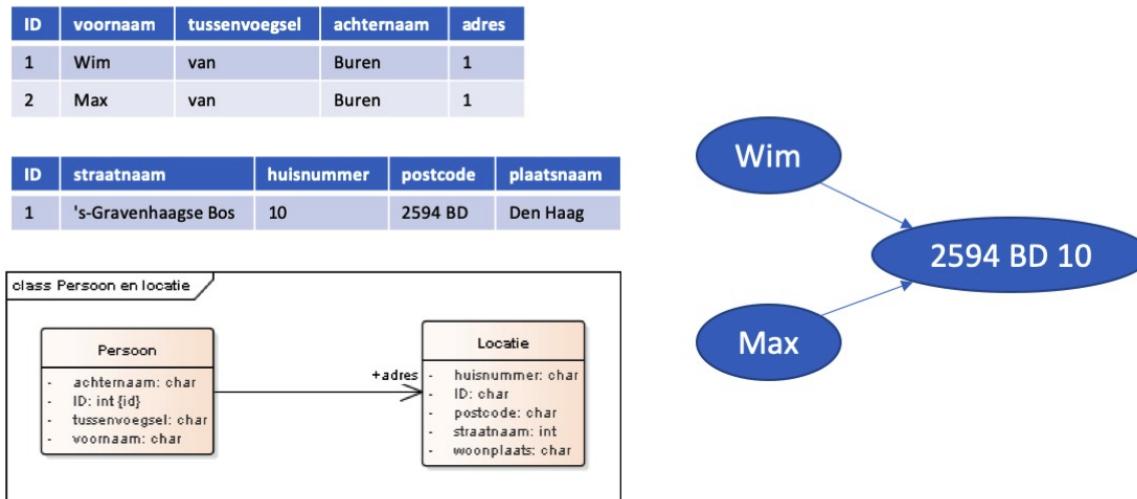
Een goed Linked Data model is juist een model dat een representatie biedt die zo dicht mogelijk bij de te beschrijven werkelijkheid ligt. Hoewel de term "normalisatie" vooral gebruikt wordt in relatie tot relationale databases, kun je het mechanisme van normalisatie wel goed gebruiken om te zien of een informatiemodel goed bruikbaar is om te vertalen naar een ontologie: een goed genormaliseerd informatiemodel zal tot minder uitdagingen leiden bij transformatie naar een

ontologie, dan een gedenormaliseerd model. Zie de afbeeldingen hieronder om een idee te krijgen van hoe normaalvormen de structuur en bereikbaarheid van informatie beïnvloeden.

ID	voornaam	tussenvoegsel	achternaam	straatnaam	huisnummer	postcode	plaatsnaam
1	Wim	van	Buren	's-Gravenhaagse Bos	10	2594 BD	Den Haag
2	Max	van	Buren	's-Gravenhaagse Bos	10	2594 BD	Den Haag



Figuur 2 Een model van object persoon in de eerste normaalvorm. Persoon en adres, in werkelijkheid twee verschillende dingen, staan hier in één klasse.



Figuur 3 Een model van objecten persoon en adres in de tweede normaalvorm. Persoon en adres zijn in deze vorm gescheiden.

Normalisatie is dus uiterst belangrijk voor een goed Linked Data model. Maar het normaliseren van een informatiemodel is alleen mogelijk wanneer je de betekenis van de data - de semantiek - begrijpt. Hieruit volgt dat het meestal niet mogelijk is om een automatische vertaling te doen van een informatiemodel naar een goed Linked Data model!

Merk op dat het wel mogelijk is om een gedenormaliseerd model over te zetten in een correct (gedenormaliseerd) Linked Data model. Echter gaat dit in tegen de principes van Linked Data. De gedenormaliseerde entiteitsrepresentatie is namelijk per definitie niet meer herkenbaar in de open wereld en heeft enkel waarde binnen een zeer specifieke (systeem) context, daarmee de herbruikbaarheid en de linkbaarheid van de data verminderend.

Een praktische richtlijn die hieruit volgt, is dat je wanneer je een klasse modelleert, je alleen die eigenschappen aan een klasse toevoegt die direct tot deze dingen behoren, ofwel essentiële eigenschappen zijn.

Nauw verwant is het punt dat gegevens over gegevens van een entiteit vaak op hetzelfde niveau als de gegevens over de entiteit worden geplaatst. Een veel voorkomend voorbeeld daarvan is geldigheid van gegevens. Echter, eigenschappen die bijvoorbeeld gaan over het registreren van informatie over de instantie in het systeem (door wie? wanneer? etc.) horen in een goed Linked Data model niet bij de entiteit zelf (de non-information resource), maar bij de information resource.

NOOT

Consequenties voor NEN 3610

In NEN 3610:2011 staan regels voor het opnemen van materiele en formele historie en materiele en formele levensduur. Deze regels moeten worden heroverwogen in het kader van de in Linked Data gewenste scheiding tussen information resource en non-information resource.

5.5 Verschillende type informatiemodellen

Een informatiemodel beschrijft de werkelijkheid. In de praktijk blijken hier niveaus in te bestaan, variërend van een zo getrouw mogelijke beschrijving van die werkelijkheid tot een specificatie van de wijze van vastlegging van die werkelijkheid in een database of uitwisselformaat. Veelal worden vier niveaus onderscheiden:

- Niveau 1: een **model van begrippen** waarin de werkelijkheid wordt beschreven door middel van de daarin gehanteerde begrippen en de relaties tot elkaar
- Niveau 2: een **conceptueel informatiemodel** waarin de werkelijkheid wordt beschreven door middel van de informatie die voor dit domein relevant is, onafhankelijk van de het ontwerp en de implementatie in systemen
- Niveau 3: een **logisch informatie- of gegevensmodel** waarin de werkelijkheid wordt beschreven door middel van de representatie van de informatie in de systemen en de uitwisseling tussen systemen en gebruikers
- Niveau 4: een **fysiek of technisch gegevens- of datamodel** waarin de werkelijkheid wordt beschreven door middel van de structuur en eigenschappen van de technologie die wordt gebruikt bij de opslag of uitwisseling

Een UML model wordt over het algemeen gebruikt voor het opstellen van modellen op niveau 2 en 3. Vaak is daarbij niveau 1 impliciet, bijvoorbeeld door geven van een definitie aan een modelement op niveau 2, waarmee feitelijk een definitie wordt gegeven aan een begrip op niveau 1. UML wordt zelden toegepast op niveau 4, tenzij sprake is van een specifiek profiel (bijvoorbeeld een DDL-profiel voor een relationele database), of het toepassen van MDA (Model Driven Architecture) voor het automatisch genereren van een model op niveau 4 vanuit een model op niveau 3.

Linked Data kan worden toegepast op elk van de niveaus. Bovendien geldt daarbij dat voor elk van de niveaus gebruik wordt gemaakt van dezelfde expressievorm: triples. Hierdoor is het mogelijk om direct vanuit het ene niveau te verwijzen naar het andere niveau

Bovendien geldt dat Linked Data een geuniformeerd fysiek datamodel kent. Hierdoor is het niet nodig om voor elk afzonderlijk model op niveau 3 een eigen fysiek datamodel te ontwikkelen of te genereren. Elk Linked Data model is een model dat bestaat uit triples, en uitgewisseld of opgeslagen kan worden op basis van standaarden. De meest bekende standaarden zijn Turtle, RDF/XML en JSON-LD.

Modellen op niveau 1 worden over het algemeen in Linked Data uitgedrukt op basis van SKOS. Ook geldt dat een goede ontologie in Linked Data zowel bruikbaar is als conceptueel informatiemodel EN als logisch informatiemodel. Dit is mogelijk, omdat een ontologie in Linked Data niet zozeer een structuur beschrijft die opgeslagen dan wel uitgewisseld wordt, maar vooral beschrijft wat de betekenis is van de termen die worden gebruikt bij deze uitwisseling of opslag.

Om in Linked Data vervolgens te specificeren op welke manier een ontologie wordt gebruikt in een uitwisseling, kan specifiek voor één uitwisseling een SHACL structuurspecificatie worden opgesteld.

Ten aanzien van de vier hierboven genoemde niveaus kan dan ook worden geconstateerd:

- Deze vier niveau's zijn ook in Linked Data aanwezig, maar wel op een iets andere wijze ingevuld.
- Het model van begrippen wordt expliciet onderscheiden van de overige modellen.
- Het fysieke of technische datamodel is geuniformeerd: deze is wereldwijd gestandaardiseerd.
- De ontologie is zowel bruikbaar op niveau 2 en 3. Bovendien wordt in de uitwisseling van concrete data explicet gerefereerd aan deze ontologie, en verwijst de ontologie terug naar het model van begrippen.
- Om expliciet een opslagstructuur of uitwisselingsstructuur te beschrijven, wordt gebruik gemaakt van SHACL structuurspecificaties, waarbij ook weer wordt verwezen naar de ontologie.

In dit document wordt voor de transformatie uitgegaan van een UML model op niveau 2/3. Dit sluit aan op het uitgangspunt dat nu veel van dergelijke modellen aanwezig zijn, en het relevant is om daarvan ook een variant te hebben, uitgedrukt in Linked Data. Vanuit een modelleringstaak zou het echter beter zijn om te beginnen met een model op niveau 1, en vervolgens dit model verder uit te werken op de lagere niveaus, waarbij zowel een UML informatiemodel als een Linked Data ontologie onstaat vanuit een gedeeld begrippenkader op niveau 1.

5.6 Setoriëntatie in Linked Data en de relatie met de UML Class

Linked Data maakt gebruik van RDF als gegevensmodel. RDF is setgeoriënteerd, dat wil zeggen, gebaseerd op de verzamelingenleer. Een klasse in RDF (rdfs:Class, owl:Class) is een set van dingen met gedeelde eigenschappen. Klassen kunnen, vergelijkbaar met objectgeoriënteerde klassen, hierarchisch gerelateerd worden. Echter, een belangrijk verschil tussen RDF en het objectgeoriënteerde paradigma is dat alle klassen (sets) kunnen overlappen. Een ding kan tegelijkertijd tot meerdere klassen behoren. Dat wil zeggen dat het ding de eigenschappen van beide klassen draagt. Hierbij is het goed om onderscheid te maken tussen meervoudige typering en multiple inheritance(meervoudige overerving).

Bij meervoudige typering wordt een ding geklassificeerd tot meerdere klassen. Dit wordt in RDF vaak toegepast om een en hetzelfde ding te beschrijven vanuit meerdere perspectieven. Een ding kan bijvoorbeeld getypeerd worden tot de klasse van datasets (void:Dataset) en de klasse van entiteiten die een herkomst hebben (prov:Entity), omdat we deze aspecten op hetzelfde ding beschrijven. Met multiple inheritance kan ook meervoudige typering bereikt worden. Het verschil is dat de typering niet direct aangebracht wordt, maar wordt afgeleid. Zo zou je een klasse Renpaard een subklasse kunnen laten zijn van de klassen Paard en Wedrenner. Hiermee kan worden afgeleid dat een instantie van de klasse Renpaard ook een instantie is van Paard en Wedrenner en eventuele superklassen van deze klassen.

In principe is het altijd wel mogelijk om een intersectie van klassen te definiëren waarmee een meervoudig getypeerd ding enkelvoudig geklassificeerd kan worden, echter leidt dit veelal tot onnodige complexiteit.

Omdat RDF uitgaat van een open wereld, is het onmogelijk om alle instanties van een klasse van te voren te kennen. Er bestaat altijd de mogelijkheid dat er een nieuwe bewering over een ding gedaan wordt, waarmee een nieuwe klassificatie gemaakt kan worden. Deze openheid biedt extreme flexibiliteit, die nodig is om een internet van Linked Data te kunnen realiseren.

Hoewel sommige objectgeoriënteerde talen het concept van multiple inheritance ondersteunen, doet het merendeel dat niet. Meervoudige typering (zonder multiple inheritance) wordt helemaal niet ondersteund. Een verschijnsel dat daardoor veel voorkomt in informatiemodellen zijn typerende / classificerende lijsten. Een objectklasse heeft dan vaak een attribuut waarvan de naam begint met 'type' of 'soort', waarmee een extra typering van instanties kan worden gegeven. Omdat RDF setgeoriënteerd is, zijn dit soort attributen in Linked Data niet nodig; in plaats daarvan kan de instantie lid gemaakt worden van meerdere klassen.

5.6.1 UML classes en OWL classes

Er bestaat een subtel verschil tussen wat een UML class vertegenwoordigt en wat een OWL class vertegenwoordigt. Wat de oorzaak van dit verschil precies is, vereist een theoretische verhandeling die we hier achterwege laten. Echter aan de hand

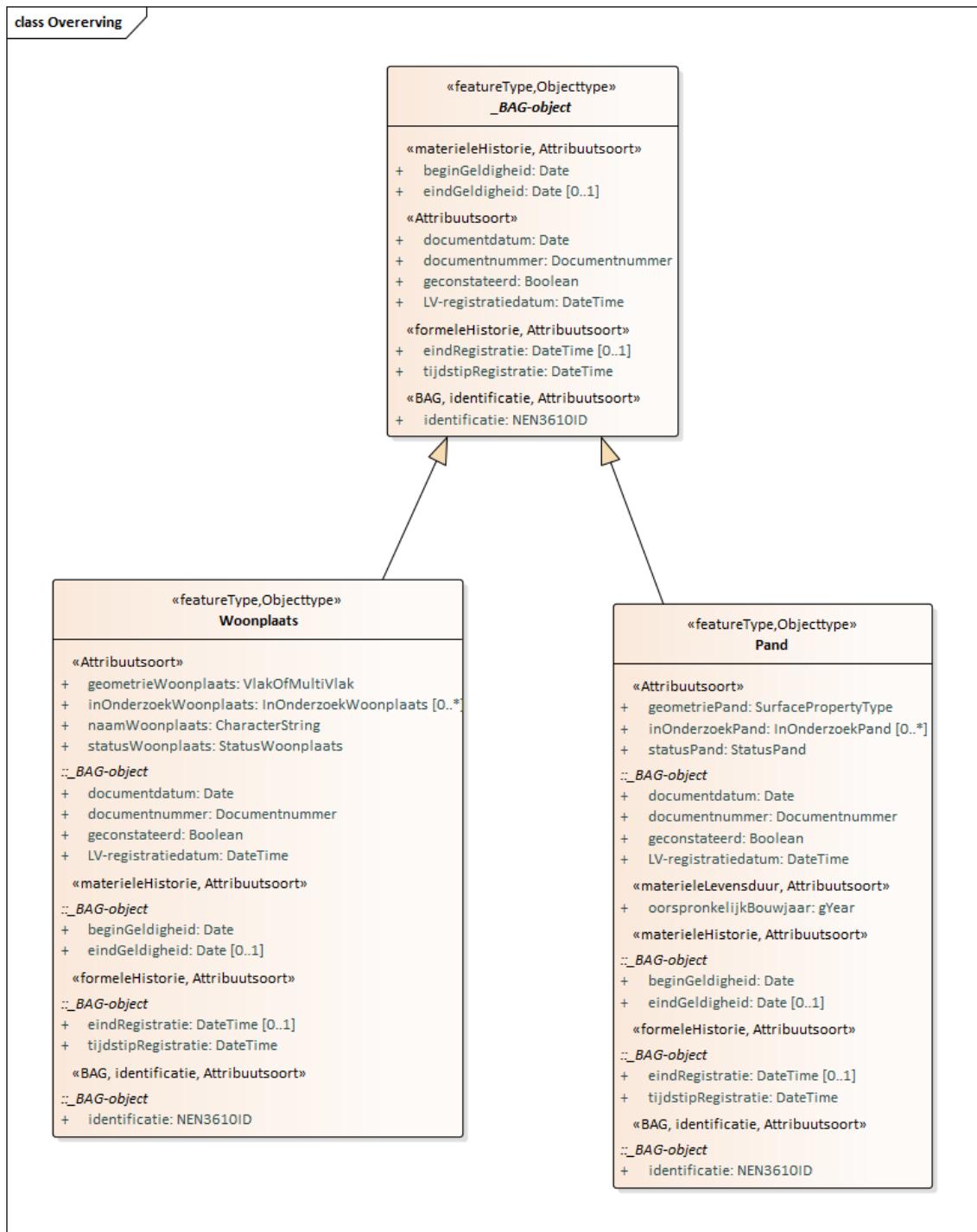
van een voorbeeld moet het lukken om hier toch enig inzicht in te verschaffen, waardoor we tijdens het transformeren tussen UML en OWL, de juiste keuzes kunnen maken.

Zowel in UML als in OWL bestaat het principe van overerving. In OWL door de [rdfs:subClassOf](#) relatie. In UML wordt overerving aangeduid met de superClass relatie. OWL gaat uitsluitend uit van semantische relaties ofwel types.

Bijvoorbeeld: een mens is een type zoogdier. We kunnen dus stellen dat de owl:Class mens een [rdfs:subClassOf](#) is van de owl:Class zoogdier. Hetzelfde is correct in UML: een UML class persoon heeft een superClass relatie met de UML class zoogdier. Dit is voor de meeste mensen intuïtief correct.

Echter, in UML mag de superclass ook voor andere doeleinden gebruikt worden. Dit wordt in de volgende paragrafen beschreven.

5.6.2 Wat is een Pand en wat is een Woonplaats



Figuur 4 Pand en Woonplaats zoals ze in IMBAG gemodelleerd zijn

Als voorbeeld nemen we een uitsnede uit IMBAG (Informatie Model Basisregistratie Adressen en Gebouwen). Als we intuïtief denken aan het concept 'Pand', dan denken we aan een gebouw, aan iets waar je naar binnen kunt lopen en dat je aan kunt raken. De formele definitie die IMBAG geeft (op het moment van schrijven) is:

Een Pand is de kleinste, bij de totstandkoming functioneel en bouwkundig-constructief zelfstandige eenheid die direct en duurzaam met de aarde is verbonden en betreedbaar en afsluitbaar is. [IMBAG]

Echter, 'Pand' overerft eigenschappen als 'documentdatum' en 'documentnummer'. Deze eigenschappen blijken overerft van de UML class BAG-object. Een andere class die van BAG-object overerft is de Woonplaats. Intuïtief denken we hier aan

een regio, met relatief veel bebouwing. De definitie volgens IMBAG:

Een Woonplaats is een door het bevoegde gemeentelijke orgaan als zodanig aangewezen en van een naam voorzien gedeelte van het grondgebied van de gemeente. [IMBAG]

5.6.3 Wat is een BAG-object?

Als we puur vanuit een semantisch perspectief (dat OWL hanteert) naar deze classes en hun overerving kijken, dan komen we tot een definitie die is samengesteld gebaseerd op het feit dat een BAG-object 'een verzameling vertegenwoordigd die zowel een woonplaatsen als een panden bevat en eigenschappen heeft als documentnummer, en documentdatum'. Dit doet vermoeden dat BAG-object dus een soort document is. Bij document denken de meeste mensen aan een stapeltje papier of misschien een pdf-je, meer generiek 'iets dat een beschrijving bevat van iets'.

5.6.4 Als je met een OWL bril naar UML kijkt: wat is een Pand?

Als we vanuit een OWL perspectief naar het UML model kijken, dan zien we onmiddellijk dat een Pand een soort document is. Als we dit samen voegen met de definitie van Pand zoals die in IMBAG wordt gegeven, dan komen we tot het volgende:

Een Pand is een soort document en is [iets dat] direct en duurzaam met de aarde is verbonden en betreedbaar en afsluitbaar is.

Dit is duidelijk een ontrecte interpretatie: een document is niet duurzaam met de aarde verbonden en niet betreedbaar.

5.6.5 Waar ontstaat het misverstand?

Wat in dit specifieke geval fout gaat is dat in IMBAG geen onderscheid wordt gemaakt tussen het Pand (het ding dat buiten staat en waar regen op kan vallen) en de documentatie dan wel registratie van dat ding. Dit onderscheid is in UML, en traditionele object-oriëntatie vaak niet nodig of niet relevant. Namelijk: alles wat in een systeem staat is al een representatie (documentatie) van iets in de werkelijkheid.

In het algemeen kunnen we zeggen dat een UML klasse een blauwdruk geeft voor de verzameling van eigenschappen en gedragingen van een instantie. Vaak betekent dit dat overerving in UML ook een semantische overerving betreft, maar niet per definitie. De UML specificatie lijkt dit te erkennen. Onder het kopje generalisatie (sectie 9.2.3.2) staat 'Type conformance' als speciaal geval genoemd:

Type conformance means that if one Type conforms to another, then any instance of the first Type may be used as the value of a TypedElement whose type is declared to be the second Type. A Classifier is a Type, and conforms to itself and to all of its generalizations. [uml]

Dus enkel in het geval dat een UML superClass voldoet aan type conformance, is de superClass relatie naar de rdfs:subClassOf relatie te vertalen. Helaas is in de meeste UML modellen niet aangeduid of het daar waar de superClass relatie is toegepast, het daadwerkelijk ook om type conformance gaat. Het is dus aan degene die de transformatie uitvoert om dit te interpreteren.

Dit verschijnsel hint op hoe OO en Linked Data verschillend naar informatie kijken. In OO declareer je een ding door te stellen ' deze auto heeft 4 deuren': Het vaststellen van het bestaan van een entiteit is onlosmakelijk verbonden van de class die geïnstantieerd wordt. In de Linked Data wereld wordt een ding gedeclareerd door te stellen 'er is een ding', dat ding van het type auto en dat ding heeft 4 deuren': de (mogelijk meervoudige, of juist volledig afwezige) classificering van een entiteit volgt pas nadat het bestaan van de entiteit is vastgesteld. Linked Data stelt ons in staat om op een later tijdstip vast te stellen dat 'dit ding is van het type speelgoed'.

NOOT

Consequenties voor NEN 3610

Er moet opnieuw gekeken worden naar de regels en aanbevelingen in NEN 3610:2011 over het hanteren van superklassen. In de basis hanteert NEN 3610 de superClass relatie voor type conformance. Echter wordt in NEN 3610 sectormodellen regelmatig het modelleerpatroon gevuld dat we in IMBAG tegenkomen: het hanteren van een abstracte superklasse die de kenmerken van alle objecten in een registratie bevat, die overerft worden door alle subklassen. Hier moet wellicht een regel of aanbeveling over worden opgenomen in NEN3610, die dit bijvoorbeeld ontraadt voor conceptuele informatiemodellen.

NEN 3610:2011 zegt niets over meervoudige overerving (waarbij een klasse meerdere superClass relaties heeft). Het zou nuttig zijn om hier wel iets over op te nemen, waarbij kan worden aangegeven dat dit in Linked Data geen probleem is.

5.7 Referentiedata

Referentiedata komt in informatiemodellen doorgaans voor als een enumeratie of codelijst, maar soms ook in de vorm van attributen of complexe datatypen. Referentiedata is een goede indicator van data die met elkaar kan worden verbonden, want het is een manier om verschillende datasets aan elkaar te koppelen of een bepaalde categorisering aan te brengen over verschillende datasets. Belangrijk om je te realiseren is dat wat voor de één referentiedata is die best in een codelijstje kan, voor de ander de data zelf is. Ofwel, referentiedata = link naar andere data.

In een Linked Data wordt referentiedata dan ook nooit gemodelleerd als een tekstueel veld (literal). In Linked Data is elke afzonderlijke referentiewaarde een verwijzing naar een resource, uitgedrukt met een IRI. Vaak betreft dit referentiedata die in een ander systeem wordt bijgehouden. Indien daarbij al gebruik wordt gemaakt van Linked Data, kan direct gebruik worden gemaakt van de IRI die in dit systeem wordt gebruikt. Indien dit niet het geval is, dan kan in het eigen systeem een IRI worden genutzt voor de betreffende referentiedata. Elementen als code en weergavenaam worden dan als eigenschappen bij de referentiedata gemodelleerd. Vaak wordt daarbij gebruik gemaakt van de SKOS vocabulaire.

VOORBEELD 1

In het informatiemodel Golfbaan (IMGolf) bestaat een enumeratie die het type hindernis omschrijft, met twee waarden:

- waterpartij
- bunker

Voor de use case van IMGolf was het blijkbaar niet nodig om hier zelfstandige klassen van te maken: van beide categorieën worden dezelfde eigenschappen vastgelegd en meer detailniveau was niet nodig. In een ander informatiemodel, dat bijvoorbeeld de informatie modelleert voor beheerders van golfbanen, kunnen Waterpartij en Bunker heel goed zelfstandige klassen zijn, gemodelleerd als subklasse van Hindernis, met eigenschappen die nodig zijn voor het beheer van waterpartijen en bunkers. Denk bijvoorbeeld aan het soort zand dat gebruikt is in de bunker. Het kan ook goed zijn dat de beheerder de volumes van deze objecten nodig heeft en dus een 3D geometrie beheert in plaats van de vlakgeometrie uit IMGolf. Soorten zand kunnen bovendien weer zelfstandige objecten zijn in een derde context...

NOOT

Consequenties voor NEN 3610

Ter overweging voor NEN 3610 is het opnemen van een aanbeveling over referentiedata. Referentiedata zijn links naar andere datasets. Als referentiedata refereert naar entiteiten, modelleer dan die entiteiten. Rationale: Wanneer deze entiteiten ook als Linked Data beschikbaar komen, is integratie van de gegevens eenvoudig.

Gebruik bijvoorbeeld blank nodes of interne URI's (URN's of UUID), om de data voorlopig te representeren als het externe data betreft. Als het interne data betreft, munt dan http IRIs. Als referentiedata refereert naar concepten die enkel nodig zijn voor categorisering, gebruik dan SKOS.

5.8 Hergebruik van bestaande vocabulaires

Linked Data maakt gebruik van bestaande, al dan niet gestandaardiseerde vocabulaires (zoals bv SKOS, FOAF, DCAT, etc.), omwille van de interoperabiliteit. Je kunt klassen en eigenschappen, die al in bestaande vocabulaires gedefinieerd zijn, vrijelijk met elkaar en met je eigen vocabulaire combineren, en je data wordt meer interoperabel hoe meer je dit doet: iedereen kan immers begrijpen wat je bedoelt als je bekende vocabulaires gebruikt.

NOOT

Consequenties voor NEN 3610

In NEN 3610 kan een aanbeveling en praktische methode worden toegevoegd over het relateren van UML klassen en eigenschappen aan corresponderende klassen en eigenschappen in bestaande Linked Data vocabulaires.

6. De NEN 3610 representatie in Linked Data

(normatief)

NEN 3610 beschrijft algemene regels voor het opstellen van een [UML](#) informatiemodel, standaard modelleerconstructies en een semantisch model. In dit hoofdstuk worden deze regels, constructies en het semantisch model uitgedrukt in [RDF](#).

Deze RDF representatie wordt expliciet **niet** geïmplementeerd als nieuwe standaard. Het betreft de RDF representatie van de bestaande NEN 3610 standaard [[NEN3610](#)], waarbij zoveel mogelijk gebruik wordt gemaakt van standaard RDF vocabulaire. Dit hoofdstuk is normatief: de genoemde URI's zijn de normatief te gebruiken URI's voor NEN 3610 elementen. De inhoud van de ontologie zelf is geen onderdeel van dit hoofdstuk en ook niet normatief: de inhoud is een één-op-één kopie van de tekst uit de NEN 3610 standaard zelf. Mochten daarbij kopieefouten zijn gemaakt, dan is de NEN 3610 tekst leidend.

De NEN 3610 representatie in RDF bestaat uit de volgende onderdelen:

- *Verwijzingen*: beschrijving van de normatieve en bibliografische verwijzingen die in de NEN 3610 standaard worden gebruikt. Als vocabulaire wordt met name Dublin Core gebruikt.
- *Begrippen*: termen en definities zoals beschreven in de NEN 3610 standaard, met name uit hoofdstuk 3 en 7. Als vocabulaire wordt met name [SKOS](#) gebruikt.
- *Klassen en eigenschappen*: de klassen en eigenschappen zoals beschreven in de NEN 3610 standaard, hoofdstuk 7. Als vocabulaire wordt met name [RDFS](#) en [OWL](#) gebruikt.
- *Gegevensregels*: regels met betrekking tot het gebruik van bovenstaande klassen en eigenschappen in een geo-Linked Data dataset, afgeleid uit de gegevensregels zoals beschreven in de NEN 3610 standaard, hoofdstuk 7. Als vocabulaire wordt [SHACL](#) gebruikt.

6.1 Verwijzingen

Hoofdstuk twee van de NEN 3610 standaard geeft de normatieve verwijzingen naar andere standaarden. Daarnaast kent de standaard nog bibliografische verwijzingen en verwijzen we vanuit het Linked Data model terug naar de NEN 3610 standaard zelf.

Elk van deze verwijzingen zijn opgenomen in het NEN 3610 Linked Data model, zodat je ook kan verwijzen naar documenten die niet op het web staan. Bij het terugverwijzen naar de standaard, verwijzen we precies naar de sectie in de standaard. Alleen die onderdelen die we ook daadwerkelijk gebruiken als bron zijn opgenomen.

Alle URI's voor verwijzingen beginnen met <http://definities.geostandaarden.nl/nen3610/id/document/>

Een volledig overzicht is te vinden op: <https://definities.geostandaarden.nl/doc/referenties/nen3610>. Een volledige [turtle](#) download is [hier](#) te vinden.

De volgende vocabulaire zijn gebruikt:

Prefix	Namespace	Vocabulaire
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework 1.1
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema 1.1
dc	http://purl.org/dc/elements/1.1/	Dublin Core (elements)
dcterms	http://purl.org/dc/terms/	Dublin Core (terms)

We maken gebruik van de volgende eigenschappen en klassen uit deze vocabulaire:

Term	Gebruik
dcterms:BibliographicResource	Elke verwijzing is een voorkomen van de klasse dcterms:BibliographicResource, een boek, een artikel of een andere brondocument.
rdf:type	Geeft het type aan van de verwijzing, in ons geval altijd dctypes:Text
rdfs:label	Elke Linked Data resource heeft een voor mensen leesbaar label. Dit is de naam van de verwijzing zelf, zoals gebruikt in de standaard
dc:title	De titel van de verwijzing, de volledig uitgeschreven titel van de verwijzing zoals opgenomen in de standaard
dcterms:isPartOf	Voor verwijzingen naar secties, nemen we ook een relatie op tussen de sectie en zijn bovenliggende sectie (zo is sectie 7.1 onderdeel van hoofdstuk 7, en hoofdstuk 7 weer onderdeel van de standaard NEN 3610:2011 als geheel)

6.2 Begrippen

Hoofdstuk drie van de NEN 3610 standaard geeft de termen en definities die gelden voor toepassen van de NEN 3610 standaard. Elke term en definitie is in het NEN 3610 Linked Data model opgenomen als skos:Concept.

Hoofdstuk zeven van de NEN 3610 standaard beschrijft de basismodellen, in het bijzonder wordt in sectie 7.3 het semantisch model beschreven. Dit model geeft aanvullende terminologie en definities. Ook deze zijn in het NEN 3610 Linked Data model opgenomen als skos:Concept.

De URI voor het begrippenkader zelf begint met <http://definities.geostandaarden.nl/id/begrippenkader/>.

Alle URI's voor begrippen beginnen met <http://definities.geostandaarden.nl/nen3610/id/begrip/>.

Een volledige beschrijving is te vinden op: <https://definities.geostandaarden.nl/doc/begrippenkader/nen3610>. Een volledige turtle download is [hier](#) te vinden.

De volgende vocabulaires zijn gebruikt:

Prefix	Namespace	Vocabulaire
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework 1.1
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema 1.1
dcterms	http://purl.org/dc/terms/	Dublin Core (terms)
skos	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System

We maken gebruik van de volgende eigenschappen en klassen uit deze vocabulaires:

Term	Gebruik
skos:ConceptScheme	Elke begrip is onderdeel van het NEN 3610 begrippenkader. Dit begrippenkader is van het type skos:ConceptScheme.
skos:Concept	Elke begrip is van het type skos:Concept. Een begrip bestaat uit een term en zijn definitie.
rdf:type	Geeft het type aan van het begrip (skos:Concept) of begrippenkader (skos:ConceptScheme).
rdfs:label	Elke Linked Data resource heeft een voor mensen leesbaar label. Dit is de term waarmee naar het begrip wordt verwezen, of de naam van het begrippenkader
skos:prefLabel	De voorkeursterm om naar het begrip te verwijzen. Meestal is zowel een term in het Nederlands als in het Engels opgegeven.
skos:altLabel	Een alternatieve term, synoniem voor het begrip.

skos:inScheme	Geeft aan tot welk begrippenkader het begrip behoort. In ons geval altijd het NEN 3610 begrippenkader.
skos:definition	De definitie van het begrip, zoals beschreven in de standaard.
skos:scopeNote	Een toelichting, voor zover aanwezig in de standaard.
skos:editorialNote	Een redactionele opmerking, waarin de rationale is opgenomen wat de reden is voor het op deze wijze beschrijven van het betreffende begrip
dcterms:source	De verwijzing naar een brondocument waaruit de definitie is gehaald. In ieder geval wordt altijd een verwijzing opgegeven naar de sectie in de originele standaard. In enkele gevallen wordt in de standaard zelf ook nog verwezen naar een andere bron. In dit geval is deze verwijzing ook opgenomen.
skos:broader	Een bovenliggend, algemener begrip. Met behulp van deze eigenschap wordt de hierarchie in het semantisch model uitgewerkt (zo is een Geo-object een breder, algemener begrip dan een Gebouw)
skos:related	Een expliciete verwijzing naar een ander begrip, voor zover dit uit de definitie blijkt

6.3 Klassen en eigenschappen

Hoofdstuk zeven van de NEN 3610 standaard beschrijft de basismodellen. Het geeft de klassen en eigenschappen die gebruikt kunnen worden in informatiemodellen die gebaseerd zijn op de NEN 3610 standaard.

Voor het realiseren van Linked Data modellen die op de NEN 3610 standaard zijn gebaseerd, is dit het meest belangrijke onderdeel. Voor Linked Data geldt dat zoveel mogelijk gebruik wordt gemaakt van algemeen gebruikte vocabulaires en ontologieën. Met dit onderdeel van het NEN 3610 Linked Data model, wordt een dergelijke ontologie gepubliceerd.

Modelleurs van Linked Data modellen die gebaseerd zijn op NEN 3610 *MOETEN* gebruik maken van deze klassen en eigenschappen, door rechtstreeks deze klassen en eigenschappen te gebruiken, of door aan te geven dat een eigen klasse of eigenschap een subklasse c.q. subeigenschap is van een klasse c.q. eigenschap uit deze ontologie.

De URI voor zowel klassen als eigenschappen begint met <http://definities.geostandaarden.nl/def/nen3610#>.

Daarbij geldt dat de verwijzing naar een klasse altijd begint met een hoofdletter (UpperCamelCase) en de verwijzing naar een eigenschap altijd begint met een kleine letter (lowerCamelCase)

Een volledige beschrijving is te vinden op: <https://definities.geostandaarden.nl/def/nen3610>

De volgende vocabulaires zijn gebruikt:

Prefix	Namespace	Vocabulaire
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework 1.1
rdfs	http://www.w3.org/2000/01/rdf-schema#	RDF Schema 1.1
dcterms	http://purl.org/dc/terms/	Dublin Core (terms)
skos	http://www.w3.org/2004/02/skos/core#	Simple Knowledge Organization System
owl	http://www.w3.org/2002/07/owl#	Web Ontology Language

We maken gebruik van de volgende eigenschappen en klassen uit deze vocabulaires:

Term	Gebruik
owl:Ontology	Alle klassen en eigenschappen zijn onderdeel van een <u>ontologie</u> , in ons geval de NEN 3610 Vocabulaire
owl:Class	Alle klassen uit de NEN 3610 standaard zijn getypeerd als owl:Class

owl:DatatypeProperty	Alle attribuut-eigenschappen uit de NEN 3610 standaard zijn getypeerd als owl:DatatypeProperty
owl:ObjectProperty	Alle associatie-eigenschappen (associaties tussen twee klassen) uit de NEN 3610 standaard zijn getypeerd als owl:ObjectProperty
rdf:type	Geeft het type aan van de vocabulaire (owl:Ontology), klasse (owl:Class) of eigenschap (owl:DatatypeProperty of owl:ObjectProperty)
rdfs:label	Elke Linked Data resource heeft een voor mensen leesbaar label. Dit is de term waarmee naar de vocabulaire, klasse of eigenschap wordt verwezen in het model.
rdfs:subClassOf	Geeft aan dat een klasse een subklasse is van een andere klasse. Dit wordt met name voor het semantisch model toegepast. Zo is een Gebouw een subklasse van een Geo-object
dcterms:subject	Geeft de relatie tussen een klasse uit het basismodel, en een begrip uit het NEN 3610 begrippenkader. Zo wordt de definitie van een klasse opgenomen bij het begrip, en volstaat daarmee een verwijzing naar het begrip voor de definitie van de klasse
skos:definition	Voor eigenschappen waarvoor geen overeenkomstig begrip is gedefinieerd, wordt hiermee de definitie van de eigenschap beschreven
skos:scopeNote	Voor eigenschappen waarvoor geen overeenkomstig begrip is gedefinieerd, kan hiermee een toelichting op de definitie van de eigenschap worden beschreven

Om de NEN 3610 klassen een zo breed mogelijk toepassingsgebied te geven, verbinden we de NEN 3610 klassen met andere standaarden. Uitgangspunt daarbij is dat de NEN 3610 klassen specifieker zijn dan de standaarden waarmee we verbinden. Op deze wijze kan een model uitgedrukt in NEN 3610 ook "gelezen" worden in een bredere context:

Prefix	Namespace	Vocabulaire
geosparql	http://www.opengis.net/ont/geosparql#	Geographic Query Language for RDF Data
schema	http://schema.org/	Schema.org

Voor de verbinding van de NEN 3610 klassen met de internationale standaarden, verbinden we het NEN 3610 model met de [OGC GeoSparql](#) Linked Data vocabulaire. We geven daarbij aan dat een nen3610:GeoObject een subklasse is van de geosparql:Feature klasse.

Voor de aansluiting met search engines, verbinden we het NEN3610 model met schema.org. Daarbij zijn twee classes relevant: [schema:Place](#) en [schema:AdministrativeArea](#). Niet elk nen3610:GeoObject is echter een schema:Place. Zo kan een trein wel gezien worden als een nen3610:GeoObject, maar niet als een schema:Place. Alle huidige NEN 3610 specialisaties van nen3610:GeoObject kunnen echter wel gezien worden als een rdfs:subClassOf schema:Place, waarbij een nen3610:RegistratieGebied gezien kan worden als een rdfs:subClassOf schema:AdministrativeArea.

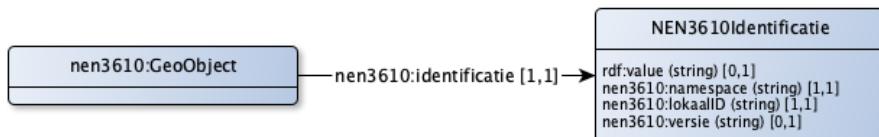
De aansluiting met de Europese Core Vocabularies [[semic-core](#)], in het bijzonder de Core Location Vocabulary verloopt niet rechtstreeks met een NEN 3610 klasse, maar indirect via geosparql. De Core Location Vocabulary gaat over adressen en geometrieën, terwijl NEN 3610 gaat over geo-objecten: dingen met een geometrie. Doordat we een geo-object hebben gedefinieerd als een speciaal soort [geosparql:Feature](#), en een dergelijk feature een geometrie kent ([geosparql:Geometry](#)), is daarmee de aansluiting gemaakt.

6.4 RDF Gegevensregels

Bij het opstellen van een Linked Data model op basis van NEN 3610 dient de opsteller zich te houden aan de gegevensregels in deze sectie. Deze gegevensregels zijn afgeleid van de gegevensregels uit sectie 7.2 van de NEN 3610 standaard. De gegevensregels zijn toegesplitst op de RDF structuur van de standaard, die vanwege de aard van Linked Data op punten anders is dan de UML standaard.

6.4.1 URI template en NEN3610ID

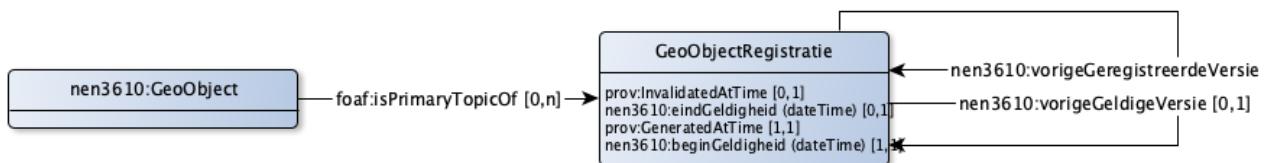
Indien een instantie van een klasse uniek identificeerbaar moet zijn binnen het domein van NEN 3610 dan moet deze klasse de eigenschap `nen3610:identificatie` hebben die verwijst naar een object met de structuur van een NEN3610ID shape. Dit is afgebeeld in onderstaand figuur en opgenomen bij de gegevensregels die als afzonderlijk bestand gebruikt kunnen worden om te valideren of een model voldoet aan de NEN3610 gegevensregels: [nen3610-shapes.ttl](#)



Omdat elk geo-object in RDF sowieso een unieke identificatie heeft op basis van een URI, is in RDF het NEN3610ID feitelijk redundant. Het is opgenomen om compliant te zijn aan de originele standaard en om een mogelijkheid te hebben om expliciet de NEN3610ID structuur over te nemen. De URI dient afgeleid te zijn van het NEN3610ID. Indien de `namespace`-eigenschap uit de NEN 3610-identificatie al een URI-namespace is, kan als het sjabloon `{namespace}{lokaalID}{versie}` worden gebruikt. Indien dit niet het geval is, dan mag gebruik worden gemaakt van een http-prefix zodat een correcte URI ontstaat, bijvoorbeeld conform het sjabloon `http://{domeinnaam}{optioneel-pad}/id/geo-object/{namespace}{lokaalID}{versie}`. De `rdf:value` mag gebruikt worden om het NEN3610ID als één string te beschrijven. Daarbij dient de waarde opgebouwd te zijn volgens het sjabloon `{namespace}{lokaalID}{versie}`

6.4.2 Temporele kenmerken en versies

In het NEN 3610 UML model wordt geen expliciet onderscheid gemaakt tussen het geo-object zelf (het fenomeen in de werkelijkheid) en de beschrijving van het geo-object (de geregistreerde eigenschappen). Dit leidt ertoe dat eigenschappen van het geo-object, zoals bijvoorbeeld de identificatie, geometrie of een naam in het UML model bij dezelfde klasse staan als de eigenschappen van de registratiemetadata, zoals de versie-eigenschappen beginGeldigheid en eindeGeldigheid. In het Linked Data model zijn deze eigenschappen ondergebracht bij de "eigen" klasse (zie voor meer uitleg hierover secties [5.3](#) en [5.4](#)). Daarbij kunnen we grotendeels gebruiken maken van de standaard PROV-O vocabulaire, zoals is afgebeeld in onderstaand figuur.



Partijen die Linked Data dataset publiceren die gebaseerd zijn op de NEN 3610 standaarden *MOETEN* gebruik maken van bovenstaand mechanisme indien zij historie willen modelleren conform de betekenis van NEN 3610.

De eigenschappen `prov:generatedAtTime` en `prov:invalidatedAtTime` komen daarbij overeen met de formele historie-eigenschappen `tijdstipRegistratie` en `eindRegistratie`.

In het model is niet expliciet de levensduur-eigenschappen opgenomen. Deze zijn af te leiden uit de afzonderlijke eigenschappen en uit de optionele relatie tussen GeoObjectRegistraties.

6.5 Gebruik van het NEN 3610 Linked Data model

Partijen die Linked Data datasets publiceren die gebaseerd zijn op de NEN 3610 standaard *MOETEN* daarbij de volgende regels in acht nemen:

6.5.1 Maak gebruik van de NEN 3610 begrippen

Indien een eigen begrippenkader wordt gehanteerd, *MOET* daarbij verwezen worden naar de NEN 3610 begrippen waar dit van toepassing is. Daarbij *MOET* gebruik worden gemaakt van de juiste eigenschappen uit de SKOS vocabulaire.

Onderstaand voorbeeld geeft aan hoe een begrip uit IM-Golf op de juiste wijze verwijst naar een NEN 3610 begrip

voorbeeld RDF representatie

```
@prefix nen3610-begrip: <http://definities.geostandaarden.nl/nen3610/id/concept/>.
@prefix imgolf-begrip: <http://definities.geostandaarden.nl/imgolf/id/concept/>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.

imgolf-begrip:Golfbaan a skos:Concept;
    skos:broadMatch nen3610-begrip:FunctioneelGebied;
    skos:definition "Een golfbaan is een functioneel gebied waar de sport golf wordt gespeeld."@nl.
.
```

6.5.2 Maak gebruik van de NEN 3610 vocabulaire

De klassen en eigenschappen die gebruikt worden in de Linked Dataset *MOETEN* direct of indirect verwijzen naar de klassen en eigenschappen van de NEN 3610 vocabulaire, voor zover van toepassing.

Voor klassen zal meestal gelden dat een subklasse is gedefinieerd, voor eigenschappen ligt meer voor de hand om rechtstreeks gebruik te maken van de NEN 3610 eigenschappen.

Onderstaand voorbeeld geeft aan hoe een voorkomen en een klasse uit IM-Golf op de juiste wijze verwijst naar de NEN 3610 vocabulaire.

In dit voorbeeld is ook de relatie gelegd tussen de klasse **imgolf:Golfbaan** en het begrip "Golfbaan". Een goede gewoonte is om hiervoor **dct:subject** te gebruiken.

Tenslotte laat dit voorbeeld ook zien hoe je vanuit eigen data rechtstreeks kunt verwijzen naar data van een ander. Een goede gewoonte is om hiervoor de URI uit de data van de andere te gebruiken als deze beschikbaar is.

voorbeeld RDF representatie

```

@prefix nen3610: <http://definities.geostandaarden.nl/def/nen3610#>.
@prefix imgolf: <http://definities.geostandaarden.nl/def/imgolf#>.
@prefix golfbaan: <http://definities.geostandaarden.nl/nen3610/imgolf/voorbeeld/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix geosparql: <http://www.opengis.net/ont/geosparql#>.
@prefix dct: <http://purl.org/dc/terms/#>.

imgolf:Golfbaan a owl:Class;
  rdfs:label "Golfbaan"@nl;
  rdfs:subClassOf nen3610:FunctioneelGebied;
  dct:subject imgolf-begrip:Golfbaan;
.

imgolf:naam a owl:DatatypeProperty;
  rdfs:label "naam"@nl;
  rdfs:subPropertyOf rdfs:label
.

golfbaan:KoninklijkeHaagseGenCC a imgolf:Golfbaan;
  owl:sameAs <http://brt.basisregistraties.overheid.nl/top10nl/id/functioneel-gebied/128367769>;
  geosparql:hasGeometry geometrie:KoninklijkeHaagseGenCC;
  imgolf:naam "Koninklijke Haagse Golf en Country Club"@nl;
  nen3610:beginGeldigheid "1938"^^xsd:Date
.
```

6.6 Metadata van datasets

Hoofdstuk 10 van de NEN 3610 behandelt metadata. Onder metadata wordt in deze norm verstaan: informatie die ruimtelijke datasets en datasetseries beschrijft die het mogelijk maakt om deze te zoeken, te evalueren en te gebruiken. Dit is een beperkte definitie omdat hij alleen betrekking heeft op datasets en datasetseries. Metadata moet voldoen aan het Nederlandse profiel op NEN-EN-ISO 19115 voor Geografie.

Voor een Linked Data model betekent dit dat voor metadata gebruik gemaakt moet worden van de [[geodcat-ap](#)] standaard, de Linked Data invulling voor de ISO 19115 standaard voor het beschrijven van metadata over geometrische datasets.

Concreet betekent dit dat elke dataset of datasetserie als **dcat:Dataset**-klasse kan worden beschreven, waarbij de metadata wordt geregistreerd via de eigenschappen van instanties van deze klasse.

NEN 3610 geeft geen specifieke versie aan van bovengenoemde standaarden. Concreet betekent dit dat verwacht wordt dat de meest recente standaard wordt gebruikt. Op het moment van schrijven wordt gewerkt aan een vernieuwing van de DCAT standaard en bijbehorende GeoDCAT profiel. Geadviseerd wordt om deze vernieuwing goed te volgen en tijdig over te stappen op de meest recente versie.

7. Transformatie van een NEN 3610-UML model naar Linked Data

Dit onderdeel is niet normatief.

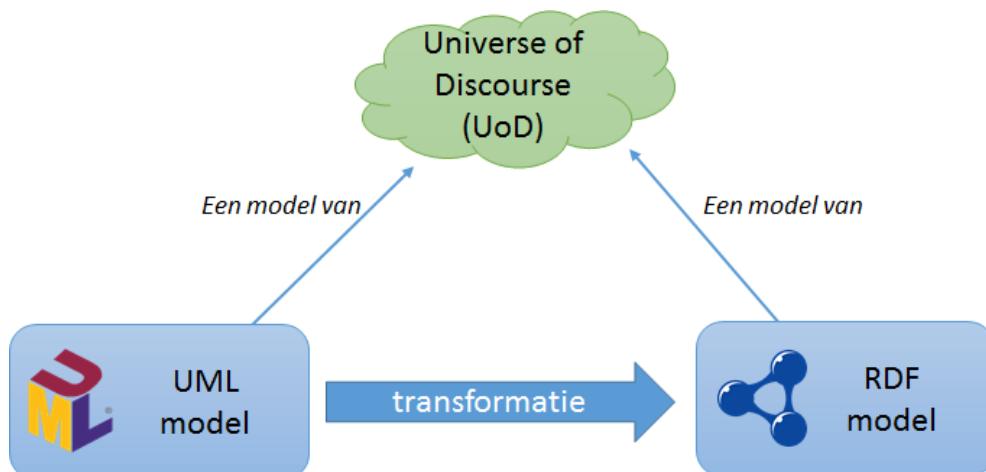
7.1 Doel van de transformatie en methode

Het doel van de transformatie voor NEN 3610 gaat er vanuit dat er op dit moment een grote hoeveelheden NEN 3610 UML modellen bestaan, waarvoor het nuttig zou zijn om van deze modellen ook een Linked Data variant te hebben. Indien een dergelijk Linked Data model zou bestaan, zou het mogelijk kunnen zijn om bestaande data conform deze modellen (bijvoorbeeld concrete data over een BAG pand, WOZ object of IMKL netwerk) te transformeren naar hun Linked Data representant.

In de transformatie wordt daarbij dan ook uitgegaan van een UML informatiemodel. Vanuit een modelleringaanpak zou het echter beter zijn om te beginnen met een gemeenschappelijk model van begrippen, en vervolgens dit model verder uit te werken naar zowel een UML informatiemodel als een Linked Data ontologie. Dit is echter niet het doel geweest van de transformatiestappen die in dit hoofdstuk zijn beschreven.

7.1.1 Correct versus juist: handmatige vertaling is deels noodzakelijk

De methode voor de toe te passen transformatie van een NEN 3610-UML bronmodel naar een Linked Data doelmodel (RDF/OWL/SHACL) bevat een aantal hoofdlijnen. De transformatieregels worden gespecificeerd conform informatie-elementen uit het metamodel van NEN 3610. Het volgt de basiselementen klassen, attributen en associaties en de daaraan gerelateerde stereotypen. Een groot deel van de transformatieregels zijn standaard op te stellen. Hier is ook al veel werk in verricht onder andere te vinden in INSPIRE RDF guidelines [[INSPIRE RDF](#)]. We noemen dit ‘standaard transformatieregels’. Naast deze standaard transformatieregels zijn er nog ‘specifieke transformatieregels’. Deze zijn specifiek omdat ze extra aanvullingen en aanpassingen zijn om er zinvolle Linked Data modellen van te maken. Ze zijn ook specifiek omdat de verschillende modelleerstijlen van Omgevingswet-DSO, de Bouwsector en Stedelijkwater-GWSW-OroX, specifieke aanpassingen vereisen. Voor elk van deze stijlen worden de standaard transformatieregels aangevuld met specifieke regels. Het geheel van standaard – en specifieke transformatieregels geeft voor elke modelleerstijl een handvat om een NEN 3610-UML model om te zetten naar het specifieke NEN 3610-Linked Data doelmodel.

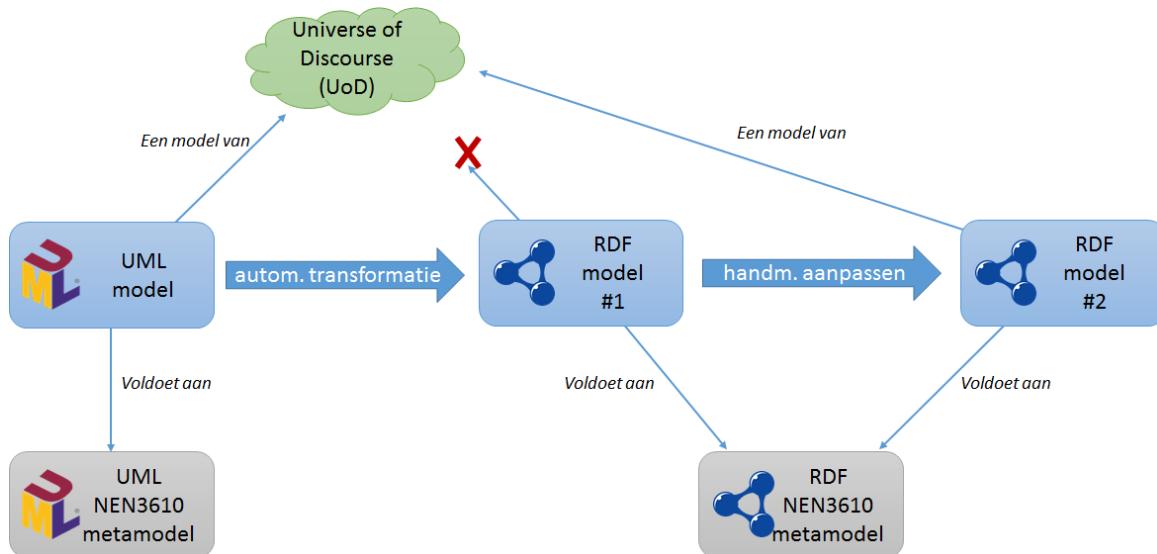


Figuur 5 UoD requirement

Bovenstaand figuur geeft ons transformatiedoel weer. Zowel een UML model als een RDF model representeren een bepaalde universe of discourse (UoD), het deel van de wereld dat we wensen te beschrijven met ons model. Waarbij het model een vereenvoudigde weergave is van deze werkelijkheid. Ons transformatiedoel is geslaagd als het RDF model dat ontstaat vanuit de transformatie van het UML model een model is van dezelfde UoD als die van het originele UML model.

We verwachten daarbij dat we dit transformatiedoel niet volledig geautomatiseerd kunnen behalen. Daartoe verschillen de uitgangspunten van UML en RDF te veel, zoals beschreven in [hoofdstuk 5](#). We wensen een *correct* model automatisch te kunnen vertalen, waarna we handmatig tot een *juist* model kunnen komen:

- Een *correct* model betekent dat het RDF model voldoet aan alle constructie-eisen van een RDF model. Het model voldoet aan het metamodel dat we in dit document vaststellen.
- Een *juist* model is een correct model waarbij het RDF model een model is van hetzelfde UoD als het originele UML model.



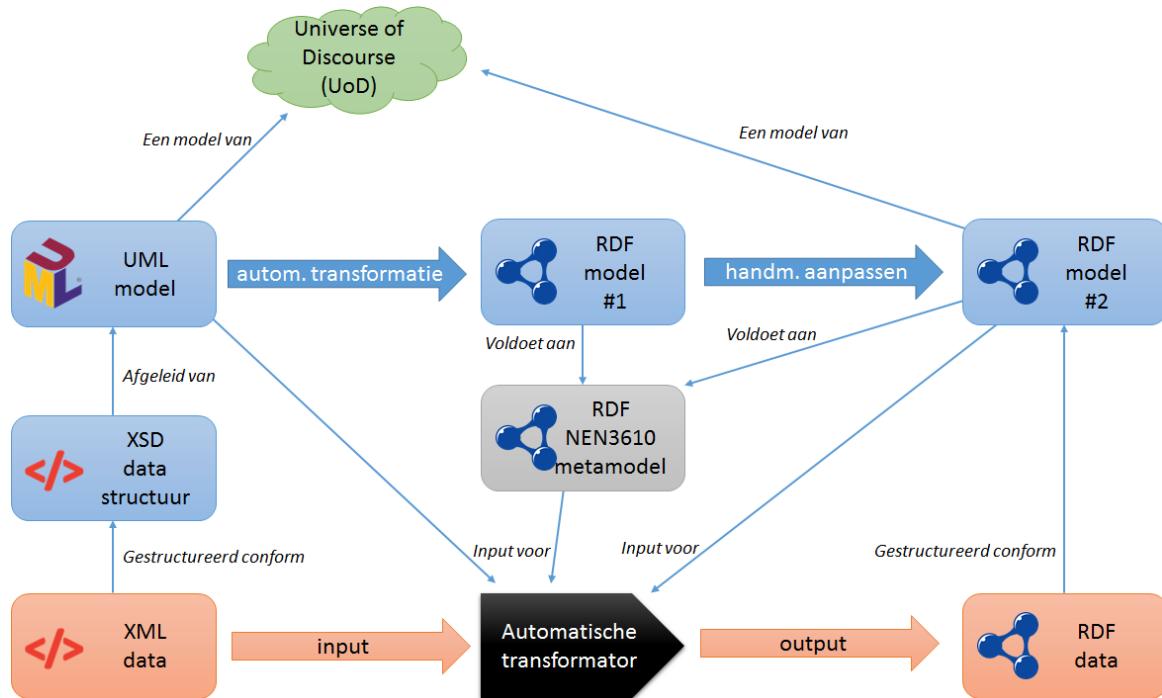
Figuur 6 Correct vs Juist model

7.1.2 Geautomatiseerde data-transformatie

Op basis van een dergelijk correct en juist model, zal het dan uiteindelijk mogelijk zijn om concrete geometrische data die gebaseerd is op het UML model, om te zetten naar RDF. Deze vertaling zou geautomatiseerd kunnen worden, waarbij zowel het originele UML model, het RDF metamodel en het correct en juiste RDF model, dat afgeleid is van het originele UML model, als input dienen.

NOOT

Na vaststelling van deze standaard kan, als een volgende stap, de automatische vertaler geïmplementeerd worden in bijvoorbeeld een XSLT stylesheet die GML data kan omzetten naar Linked Data.



Figuur 7 Transformatie van model en data

7.1.3 Geen roundtrip

Het doel van de transformatieregels is om hetzelfde UoD te beschrijven. Hiervoor is van belang dat de betreffende informatie hierover wordt overgenomen. Omdat UML en RDF andere uitgangspunten kennen, zal daarbij aan beide kanten informatie noodzakelijk zijn die niet per sé noodzakelijk is aan de andere kant. Hoewel deze informatie opgenomen zou kunnen worden (bijvoorbeeld als onderdeel van een UML profiel of als aanvullende eigenschappen aan de RDF resources), is dit voor de NEN 3610 vertaling niet uitgevoerd. Een roundtrip of volledige transformatie van alle informatie is dan ook niet uitgewerkt. Alleen die informatie wordt getransformeerd die noodzakelijk is om te komen tot een correct en juist model, waarbij voor dat laatste bovendien handmatige stappen noodzakelijk zijn, zoals beschreven in de voorgaande secties.

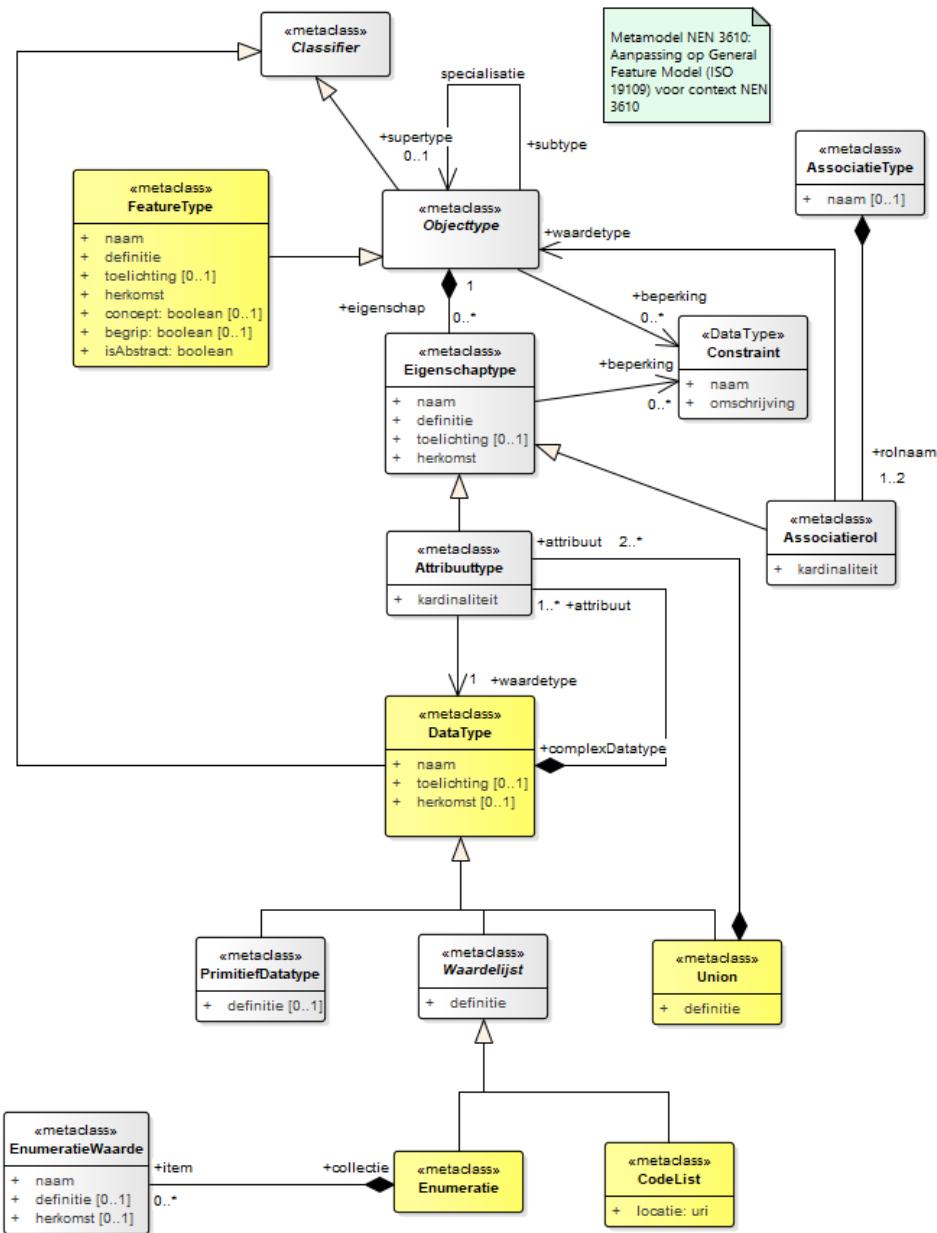
Het Metamodel Informatie Modellering [MIM11] heeft wel tot doel om een volledige transformatie en roundtrip te verzorgen. De uitwerking hiervan heeft gelijk opgelopen met de uitwerking van de NEN 3610 opzet. Het verschil is dat het MIM uitgaat van een eigen metamodel, uitgedrukt in een UML profiel en een RDF ontologie. Hierdoor is het wel mogelijk om een MIM model volledig uit te drukken in UML en in RDF, inclusief een roundtrip transformatie.

Een vergelijkbaar initiatief heeft plaatsgevonden in Vlaanderen. Hier is geen eigen metamodel ontwikkeld, maar wordt de vertaling direct vanuit Enterprise Architect opgezet [OSLO-EA-RDF].

7.2 Bronmodel: NEN 3610 metamodel

Het bronmodel in de transformatie moet voldoen aan het metamodel achter NEN 3610.

Het NEN 3610 - metamodel omschrijft de metaklassen die gebruikt worden om een informatiemodel op te bouwen. Elke metaklasse is een type informatie-element. De instanties van de metaklassen zijn de benoemde informatie-elementen in een informatiemodel. In het informatiemodel zijn ze te herkennen aan de UML conventies die voor dat element gelden of het specifiek benoemde stereotype (in geel aangegeven). Bij elke metaklasse is opgenomen wat hun eigenschappen zijn. Het metamodel van NEN 3610 volgt het General Feature Model (GFM) van ISO 19109 [iso-19109-2015]. Het in het onderstaande UML beschreven metamodel is daar een aanpassing op voor toepassing binnen de context van NEN 3610.



Figuur 8 Metamodel van NEN 3610. Aanpassing op General Feature Model van ISO 19109. In geel de als stereotype opgenomen metaklassen.

7.2.1 Toelichting op metamodel NEN 3610

Voor NEN 3610 is **Objecttype** een abstracte metaklasse. Alle objecttypen (of objectklassen) in NEN 3610 zijn feature type. Een **FeatureType** is een **Objecttype** dat geassocieerd is met een locatie. In NEN 3610 vallen die onder het semantische objecttype **GeoObject**. Een **GeoObject** heeft dus als stereotype «**FeatureType**». Dit is identiek aan de ISO191xx set van standaarden. Een **FeatureType** heeft 0 of 1 supertypen; 0 of meer eigenschappen (**EigenschapType**); 0 of meer beperkingen (**Constraint**). Eigenschappen zijn attributen (**AttribuutType**) of relaties naar andere objecttypen. Een relatie wordt gerealiseerd door een ‘uitgaande’ associatie met optioneel een naam maar verplicht de rol (**Associatieroel**) van het ‘target’ objecttype’.

Attributen hebben een waardetype dat door een datatype wordt beschreven. Een **DataType** is een **PrimitiefDatatype** (integer, characterstring boolean, gm_surface enz), een **Waardelijst**, een **Union** of gewoon een datatype. In het laatste geval is het een complex datatype dat is samengesteld uit één of meer attributen. Een **Union** faciliteert een keuze van één

attribuut uit een lijst van twee of meer. Een **Enumeratie** is een niet uitbreidbare **Waardelijst** in de namespace van het model. Een **CodeList** is een externe lijst waarvan de waarden buiten het model worden beheerd.

7.3 Transformatie: basisregels - encoding

7.3.1 Inleiding

Deze paragraaf bevat de beschrijving van de basisregels en de uitwerking van details voor transformatie van UML constructen naar Linked Data.

Voor de transformatie van een UML model naar Linked Data maken we gebruik van de volgende vocabulaires:

- RDF
- RDFS
- OWL
- SKOS
- SHACL
- GeoSparql

De reden om te kiezen voor deze vocabulaires is:

- Internationale standaarden. Deze standaarden zijn op wereldwijde schaal vastgesteld en in gebruik. Het zijn standaarden die door de W3C zijn vastgesteld, en daarmee door de organisatie die het beheer voert over de Linked Data standaarden.
- We gebruiken standaarden die specifiek zijn ontworpen voor het doel waar wij ze voor willen inzetten:
 - RDF/RDFS/OWL voor het benoemen en typeren van de termen voor klassen (owl:Class), attributen (owl:DatatypeProperty) en relaties (owl:ObjectProperty).
 - RDFS/OWL voor het specificeren van een formele ontologie (door middel van rdfs:range en rdfs:domain, rdfs:subClassOf en owl:Restriction)
 - SKOS voor het opnemen van definities en aanvullende beschrijvingen voor de gebruikte termen
 - SHACL voor het specificeren van de gegevensstructuren en gegevensregels zoals aanwezig in het UML model
- De standaarden sluiten goed aan bij de denkwijze binnen UML, waardoor een vergelijking tussen UML en Linked Data "relatief" eenvoudig is.
- Het Linked Data principe maakt het mogelijk om de vier genoemde onderdelen (terminologie, ontologie, betekenis en gegevensregels) van elkaar te onderscheiden en afzonderlijk te beheren. Dit maakt het mogelijk om specifieke aanpassingen te doen, mocht de opzet in het UML model anders geïnterpreteerd moeten worden dan uit een automatische vertaling mogelijk is. We verwachten dat dit met name zal spelen bij de formele ontologie.

Specifieke aandacht is nodig voor de vertaling van het UML enerzijds naar RDFS en OWL formele ontologie, en anderzijds naar SHACL gegevensregels.

In beginsel berust de vertaling naar de formele ontologie en naar de SHACL gegevensregels op dezelfde UML onderdelen. Zo worden UML cardinaliteiten enerzijds vertaalt naar owl:Restrictions, en anderszijds naar sh:PropertyShapes. Je zou kunnen stellen dan één van de twee voldoende is. Echter, waar in UML het onderscheid tussen formele ontologie en gegevensregels vaak impliciet is, bestaan hier in Linked Data expliciet verschillende vocabulaires voor.

Door beide varianten te realiseren, bestaat de mogelijkheid voor de afnemer om een keuze te maken welke variant het beste past bij het originele UML model en de specifieke use case.

Hieronder benoemen we enkele verschillen en overeenkomsten tussen UML, SHACL en OWL:

- Zowel UML als SHACL gaan uit van een closed-world assumption, waarbij het model *beperkt* wat mogelijk is. Dit in tegenstelling tot OWL dat uitgaat van een open-world assumption, waarbij het model *aftreidt* wat mogelijk is.
- Zowel **owl:maxCardinality 1** als **sh:maxCount 1** komen overeen met de x..1 cardinaliteit in UML. Er bestaat echter ook een verschil. Mocht bijvoorbeeld bij een eigenschap "is-getrouwde-met" gelden dat een persoon maar met één ander persoon getrouwde mag zijn, en in de database treffen we de waarden "Piet is getrouwde met Marie" en "Klaas is getrouwde met Marie", dan veronderstelt de owl restrictie dat Piet en Klaas dezelfde personen zijn, terwijl de SHACL restrictie aangeeft dat de dataset niet voldoet aan de betreffende gegevensregel.
- Enzelfde verschil speelt rondom het gebruik van **rdfs:range** en **rdfs:domain**. Indien bijvoorbeeld een property **ex:naam** wordt geïntroduceerd met **rdfs:range ex:Persoon**, dan betekent dit dat als een subject een **ex:naam** heeft, het een persoon is (zelfs als een dergelijke naam is gegeven aan iets dat eigenlijk een schip is). In geval van SHACL zou een PropertyShape gespecificeerd worden die stelt dat de eigenschap **ex:naam** alleen maar gebruikt mag worden bij de klasse **Persoon**.
- OWL is ontworpen voor classificatie-doeleinden (afleidingen), terwijl SHACL ontworpen is voor data-validatie. Zie [shacl and owl](#) voor een uitgebreide vergelijking.
- Het gebruik van SHACL maakt het mogelijk om Linked Data structuren te valideren (zie bijvoorbeeld de [SHACL playground validator](#)).
- Het gebruik van OWL maakt het mogelijk de kennis in de formele ontologie te gebruiken om nieuwe informatie af te leiden uit bestaande data. Zie [owl restrictions](#) voor een goede introductie.

Het belangrijkste criterium voor het gebruik van SHACL versus OWL formele ontologie ligt hiermee vooral in het beoogde doel van UML:

- als vastlegging van formele kennis over de ontologie te gebruiken voor het afleiden van nieuwe informatie, of
- als vastlegging van gegevensregels over de informatie die mag worden vastgelegd en mag worden uitgewisseld in berichten.

Vaak is dit onderscheid niet zo scherp te maken, waardoor wij beide varianten aanbieden. De visualisaties die in dit document staan van de RDF modellen (SHACL en/of OWL) zijn op beide manieren te lezen: als restricties, of als afleidingen.

7.3.2 Basis doelmodel: metamodel op basis van W3C standaarden



Figuur 9 Metamodel van NEN 3610 uitgedrukt als Linked Data

De kleuren in het figuur geven de onderdelen aan van het Linked Data metamodel. Het lichtgele onderdeel betreft de kern van het Linked Data metamodel. De overige onderdelen betreffen uitbreidingen op deze kern die afhankelijk van de situatie relevant zullen zijn. Dit wordt hieronder verder toegelicht.

In de Linked Data community is het gebruikelijk om voor een model zoveel mogelijk gebruik te maken van vocabulaires die reeds breed worden toegepast door anderen. Op deze manier wordt de interoperabiliteit vergroot. Voor het metamodel NEN 3610 maken we gebruik van bestaande en wereldwijd veelvuldig toegepaste standaarden. Deze standaarden zijn in beheer bij de [W3C](#), de organisatie die de webstandaarden beheert. Voor ons metamodel gaat het daarbij om de volgende standaarden:

Prefix	Standaard	Namespace
rdf	Resource description framework	http://www.w3.org/1999/02/22-rdf-syntax-ns#

rdfs	RDF schema	http://www.w3.org/2000/01/rdf-schema#
owl	Web ontology language	http://www.w3.org/2002/07/owl#
sh	Shape constraint language	http://www.w3.org/ns/shacl#
skos	Simple knowledge organization system	http://www.w3.org/2004/02/skos/core#
prov	Provenance ontology	http://www.w3.org/ns/prov#
dct	Dublin core terms	http://purl.org/dc/terms/
foaf	Friend of a friend	http://xmlns.com/foaf/0.1/

Deze vocabulaires worden op de volgende wijze ingezet:

- RDF, RDFS en OWL worden gebruikt om de terminologie te beschrijven waarin de data wordt uitgedrukt. In [UML](#) termen worden hiermee de klassen, relaties en attributen beschrijven die geïnstantieerd worden als concrete data.
- RDFS en OWL worden ook gebruikt om de formele semantiek van de data te specificeren, voor zover deze uit het UML model zijn is te leiden.
- SHACL wordt gebruikt om de gegevensregels te beschrijven die voor de data gelden. In UML termen gaat het daarbij over welke attributen horen bij welke klassen, het [datatype](#) van attributen, wat de cardinaliteiten zijn van attributen en relaties en tussen welke klassen relaties mogen liggen.
- SKOS wordt gebruikt om de betekenis te beschrijven. In UML termen gaat het daarbij om het beschrijven van de betekenis van klassen, relaties en attributen.
- PROV wordt gebruikt om de versiehistorie (provenance, herkomst) van de data te beschrijven
- Dublin Core terms en FOAF worden gebruikt voor enkele specifieke relaties die niet met één van eerder genoemde vocabulaires is in te vullen.

Met het gebruik van SKOS is het in het Linked Data [metamodel](#) mogelijk geworden om een rijkere structuur aan te brengen als het gaat om de betekenis van concepten. Hierdoor wordt feitelijk een invulling gegeven aan de standaard ISO 19126 Feature Concepts Dictionaries [[iso-19126-2009](#)].

7.3.3 Basisregels voor transformatie

algemene overwegingen, best practices, zie ook hoofdstuk 4

7.3.4 Standaard transformatieregels

transformatie regels van nen 3610 source element naar RDF Linked Data target element

7.3.4.1 Klassen

7.3.4.1.1 KLASSEN ALGEMEEN

Een klasse in een NEN 3610 UML model wordt als instantie van [owl:Class](#) gemodelleerd. Daarnaast wordt de naam van de klassen als [rdfs:label](#) eigenschap opgenomen bij de klasse. Voor elke klasse wordt ook een instantie van [sh:NodeShape](#) geïntroduceerd als aanknopingspunt voor de gegevensregels uitgedrukt in SHACL.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:Parcours a owl:Class ;
    rdfs:label "Parcours" ;
    rdfs:comment """Een Parcours is een golfbaandeel dat van een tee via de fairway
(indien aanwezig) tot de bijbehorende green loopt""";
.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Parcours a sh:NodeShape ;
    sh:targetClass imgolf:Parcours ;
.
```

Instanties van klassen worden, zoals standaard in RDF, verbonden met den klassen via een **rdf:type** eigenschap (of in Turtle met de verkort versie daarvan: **a**).

voorbeeld RDF representatie

```
:hole1 a imgolf:Parcours .
```

NOOT

In een NEN 3610 model kunnen zowel abstracte als concrete klassen voorkomen. In UML kun je daarvan afleiden dat je geen instanties mag hebben van abstracte klassen, maar alleen van concrete klassen. In RDF wordt geen onderscheid gemaakt tussen het abstract of concreet zijn van klassen. In RDF worden klassen beschouwd als sets van dingen. Als je een set kunt beschrijven, dan kunnen er ook dingen zijn die tot die set behoren.

7.3.4.1.2 KLASSE MET STEREOTYPE «FEATURETYPE»

Een NEN 3610 UML klasse met stereotype featureType wordt gebruikt om een geo-object te representeren. In Linked Data wordt eenzelfde object ook als instantie van een geo-object gezien. Daarom wordt in Linked Data een klasse met het stereotype featureType ook een klasse die een sub-type is van één van de NEN 3610 klassen. De NEN 3610 klassen zijn subklassen van **nen3610:GeoObject** welke zelf een subklasse is van de klasse Feature uit het GeoSPARQL vocabulaire (**gsp:Feature**). Hiermee kunnen alle instanties van de NEN 3610 klassen ook beschouwd en beschreven worden met het breed toegepaste GeoSPARQL vocabulaire.

voorbeeld RDF representatie

```
imgolf:Golfbaan a owl:Class ;
    rdfs:subClassOf nen3610:FunctioneelGebied ;
.

nen3610:FunctioneelGebied rdfs:subClassOf nen3610:GeoObject .
nen3610:GeoObject rdfs:subClassOf gsp:Feature .
```

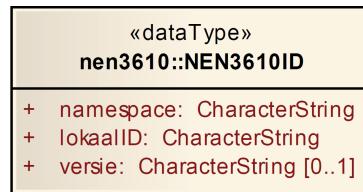
vergelijking met andere best practices

Komt overeen met de [Inspire RDF Guidelines \[INSPIRERDF\]](#).

7.3.4.1.3 KLASSE MET STEREOTYPE «DATATYPE»

Een NEN 3610 UML klasse met stereotype «datatype» wordt toegepast wanneer er sprake is van een samenstelling van attributen die een eigenschap schrijft.

Het is niet mogelijk om voor deze datatypen een standaard transformatieregel te beschrijven die in alle gevallen de semantisch correcte betekenis van het datatype uitdrukt. Dit is afhankelijk van wat het samenstel van attributen uitdrukt. Daarom wordt een klasse met stereotype «datatype» standaard getransformeerd naar een normale klasse met eigenschappen. Een voorbeeld van een datatype is het NEN3610ID datatype.



Figuur 10 Klasse met stereotype «datatype»

Indien van het getransformeerde model zowel de ontologie (owl:Class) als gegevensregels (sh:NodeShape) worden gebruikt, dan kan de owl:Class achterwege blijven indien het datatype puur als gegevensregel is bedoeld.

Daarnaast zal in de concrete data mogelijk geen identificatie (URI) nodig of beschikbaar zijn, aangezien in het originele model slechts sprake was van een datatype. In dat geval zou hiervoor een blank node gebruikt kunnen worden, of een gegenereerde URN (bijvoorbeeld: `urn:uuid:{uuid}`)

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

# Het datatype als klasse
nen3610:Identificatie a owl:Class .

nen3610:namespace a owl:DatatypeProperty ;
  rdfs:domain nen3610:Identificatie ;
  rdfs:range xsd:string ;

.

nen3610:lokaalID a owl:DatatypeProperty ;
  rdfs:domain nen3610:Identificatie ;
  rdfs:range xsd:string ;

.

nen3610:versie a owl:DatatypeProperty ;
  rdfs:domain nen3610:Identificatie ;
  rdfs:range xsd:string ;

.

#-----#
# Gegevensregels
#-----#

# Een attribuuttype met als datatype de NEN3610Identificatie
sh:property [
  sh:name "nen3610:identificatie";
  sh:path nen3610:identificatie;
  sh:node shape:NEN3610Identificatie;
];


# Het datatype als nodeshape
shape:NEN3610Identificatie a sh:NodeShape ;
  rdfs:label "Shape voor NEN3610 Identificatie" ;
  rdfs:comment "De combinatie van 'namespace' van een registratie, lokale identificatie en versie-informatie maken een object uniek identificeerbaar. Met de informatie van deze klasse kan daardoor met zekerheid worden verwezen naar het geïdentificeerde object." ;
  sh:targetClass nen3610:Identificatie ;
  sh:property [
    sh:name "nen3610:namespace" ;
    sh:path nen3610:namespace ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:pattern "^[A-Za-z0-9_,.-]*$" ;
    sh:message "namespace voldoet niet aan de eisen" ;
  ] ;
  sh:property [
    sh:name "nen3610:lokaalID" ;
    sh:path nen3610:lokaalID ;
    sh:datatype xsd:string ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:pattern "^[A-Za-z0-9_,.-]*$" ;
    sh:message "lokaalId voldoet niet aan de eisen" ;
  ] ;
  sh:property [
    sh:name "nen3610:versie" ;
  ]

```

```

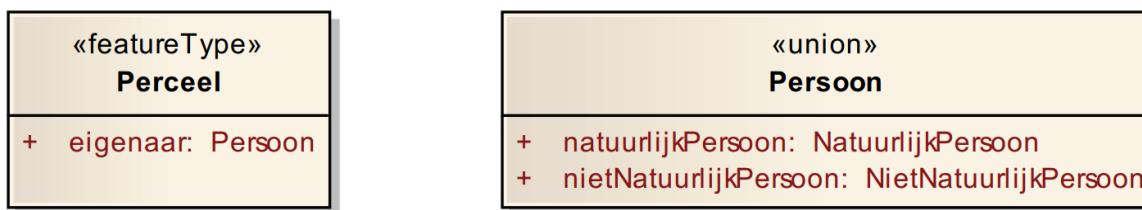
sh:path nen3610:versie ;
sh:datatype xsd:string ;
sh:maxCount 1 ;
sh:pattern "^[A-Za-z0-9_,-]*$";
sh:maxLength 25 ;
sh:message "versie voldoet niet aan de eisen" ;
] ;
sh:property [
    rdfs:comment "De gecombineerde identificatie" ;
    sh:name "rdf:value" ;
    sh:path rdf:value ;
    sh:datatype xsd:string ;
    sh:maxCount 1 ;
    sh:pattern "^[A-Za-z0-9_,-]*$";
    sh:message "value voldoet niet aan de eisen" ;
] ;
.
.
```

vergelijking met andere best practices

Komt grotendeels overeen met de [Inspire RDF Guidelines \[INSPIRERDF\]](#). De Inspire RDF Guidelines beschrijven nog een uitzondering, maar deze uitzondering laten wij buiten beschouwing.

7.3.4.1.4 KLASSE MET STEREOTYPE «UNION»

Een union is een gestructureerd datatype zonder identiteit waarvan precies één van de eigenschappen aanwezig is in elke instantie.



Figuur 11 Klasse met stereotype «union» - een union van objecten

In dit voorbeeld moet een perceel altijd een eigenaar hebben, maar kan deze eigenaar een natuurlijk persoon of een niet-natuurlijk persoon zijn.

Een klasse met stereotype union «union» kan gebruikt worden voor een union van concrete dingen, of een union van letterlijke waardes. In het geval van een union van dingen transformeren we dit naar een klasse met een eigenschap `owl:unionOf` die verwijst naar een `rdf>List` met een opsomming van de klassen van dingen die onderdeel zijn van de union in het oorspronkelijke model. In de bijbehorende SHACL nodeshape beschrijven we de gegevensregel dat de waarde van de eigenschap een instantie moet zijn van een van de klassen uit de union.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

ex:eigenaar a owl:ObjectProperty .

ex:Perceel a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty ex:eigenaar ;
    owl:allValuesFrom ex:Persoon ;
  ] ;

.

ex:NatuurlijkPersoon a owl:Class .

ex:NietNatuurlijkPersoon a owl:Class .

ex:Persoon a owl:Class ;
  owl:unionOf (
    ex:NatuurlijkPersoon
    ex:NietNatuurlijkPersoon
  ) ;

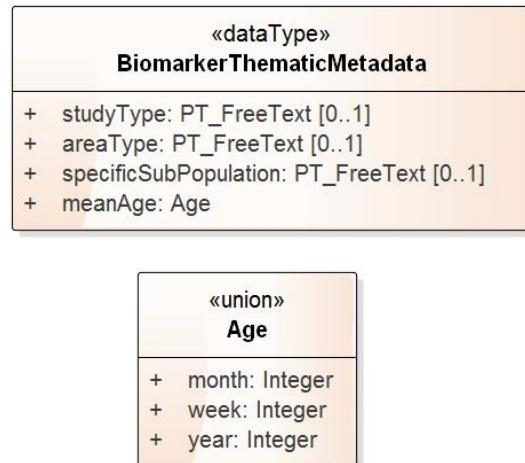
.

#-----#
# Gegevensregels
#-----#

ex-sh:Perceel a sh:NodeShape ;
  sh:targetClass ex:Perceel ;
  sh:property [
    sh:path ex:eigenaar ;
    sh:or (
      [
        sh:class ex:NatuurlijkPersoon ;
      ]
      [
        sh:class ex:NietNatuurlijkPersoon ;
      ]
    )
  ] ;
.
```

Bij een union van letterlijke waarden wordt de union niet gerepresenteerd als een union van klassen, maar een union van eigenschaprestricties per specifieke eigenschap die in de oorspronkelijke union voorkomt. De SHACL gegevensregel wordt gemodelleerd als een keuze tussen de verschillende eigenschappen in de union.

Neem het voorbeeld van een union van letterlijke waarden uit de [Inspire RDF Guidelines \[INSPIRERDF\]](#).



*Figuur 12 Klasse met stereotype «union» - een union van letterlijke waarden
bron: [\[INSPIRERDF\]](#)*

Deze wordt als volgt gemodelleerd.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

bio:meanAge a owl:ObjectProperty .

bio:BiomarkerThematicMetadata a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty bio:meanAge ;
    owl:allValuesFrom bio:Age ;
  ] ;

.

bio:month a owl:DatatypeProperty .

bio:week a owl:DatatypeProperty .

bio:year a owl:DatatypeProperty .

bio:Age a owl:Class ;
  owl:unionOf (
    [
      a owl:Restriction ;
      owl:onProperty bio:month ;
      owl:allValuesFrom xsd:string ;
    ]
    [
      a owl:Restriction ;
      owl:onProperty bio:week ;
      owl:allValuesFrom xsd:string ;
    ]
    [
      a owl:Restriction ;
      owl:onProperty bio:year ;
      owl:allValuesFrom xsd:string ;
    ]
  ) ;
.

#-----#
# Gegevensregels
#-----#


bio-sh:BiomarkerThematicMetadata a sh:NodeShape ;
  sh:targetClass bio:BiomarkerThematicMetadata ;
  sh:property [
    sh:path bio:meanAge ;
    sh:node bio-sh:Age ;
  ] ;

.

bio-sh:Age a sh:NodeShape ;
  sh:or (
    [
      sh:path bio:month ;
      sh:datatype xsd:integer ;
      sh:minCount 1 ;
      sh:maxCount 1 ;
    ]
  )

```

```
[  
    sh:path bio:week ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
]  
[  
    sh:path bio:year ;  
    sh:datatype xsd:integer ;  
    sh:minCount 1 ;  
    sh:maxCount 1 ;  
]  
)  
.
```

vergelijking met andere best practices

In de [Inspire RDF Guidelines](#) wordt voorgeschreven om een union niet als een union van klassen te modelleren, omdat dit zou kunnen leiden tot het creëren van onlogische instanties van de union-klasse, wat ongewenst is [[INSPIRERDF](#)]. Dit komt voor wanneer een union een combinatie van eigenschappen met letterlijke waardes gebruikt. Het is een terechte constatering, maar de voorgestelde alternatieve modellering is overgecompliceerd voor het merendeel van de gevallen. In de standaard transformatieregels houden wij hier dan ook geen rekening mee. Het is wel aan te raden om hier in de verdere handmatige transformatie rekening mee te houden.

7.3.4.1.5 KLASSE MET STEREOTYPE «EXTERNAL»

Indien in een model wordt verwezen naar een object in een andere registratie, kan de klasse waarnaar wordt verwezen worden opgenomen in het UML met het stereotype «external».



Figuur 13 Klasse met stereotype «external»

In een Linked Data omgeving is het gebruikelijk om gebruik te maken van bestaande ontologieën wanneer deze beschikbaar zijn. Als er een ontologie bestaat waar de externe klasse gedefinieerd is kun je direct gebruik maken van de klasse. Wanneer dit niet het geval is zal daar voor een eigen klasse geïntroduceerd moeten worden. Omdat dit per model kan verschillen worden klassen met stereotype «external» en de associaties naar deze klasse getransformeerd als een normale klasse en associatie maar met een herkenbare afwijkende namespace. Bijvoorbeeld <http://external.com/> zodat er handmatig bepaald kan worden wat de beste aanpak is voor het specifieke model.

voorbeeld RDF representatie

```

imgolf:verwijzingBAG a owl:ObjectProperty .

external:Pand a owl:Class .

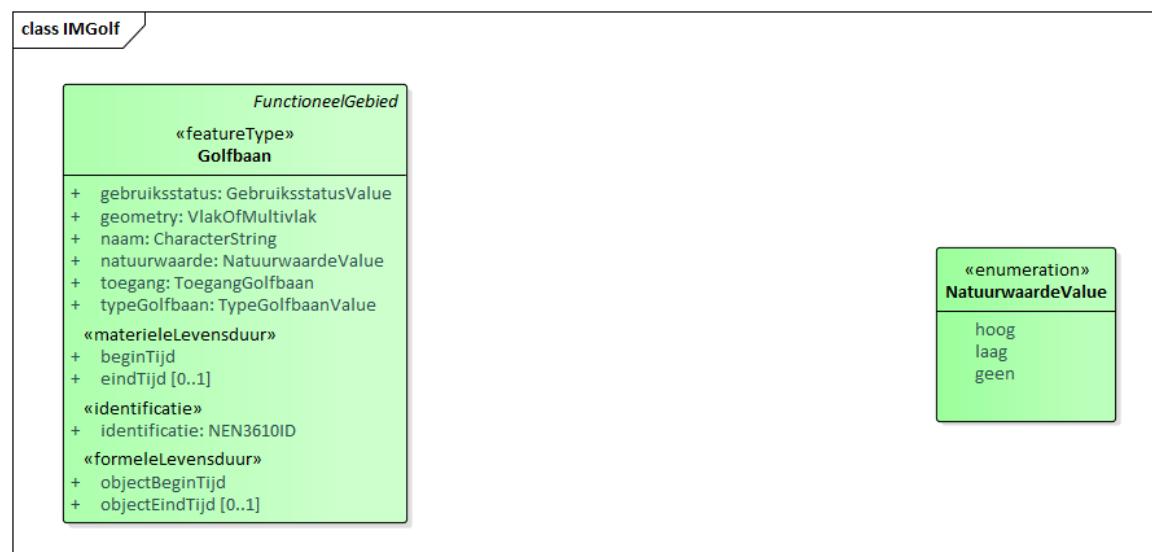
imgolf:Gebouw a owl:Class ;
  owl:subClassOf [
    a owl:Restriction ;
    owl:onProperty imgolf:verwijzingBAG ;
    owl:allValuesFrom external:Pand ;
  ] ;
.
.
```

vergelijking met andere best practices

Geen.

7.3.4.1.6 KLASSE MET STEREOTYPE «ENUMERATION»

Een enumeratie is een klasse die een lijst van waarden weergeeft. Deze kan worden gebruikt op plaatsen waar voor een bepaalde waarde uit een beperkt aantal vooraf bekende mogelijkheden behoort te worden gekozen.



Figuur 14 Klasse met stereotype «enumeration»

Een enumeratie kan verschillende soorten dingen opsommen. Een lijst met waardes, bijv. een opsomming van nummers, maar ook een lijst met concepten, datatypes, of objecten. Het is dan ook niet triviaal om een goede automatische vertaling te bepalen die een enumeratie kan vertalen naar Linked Data. Om deze reden kiezen we voor een standaardtransformatie naar een klasse gelijknamig aan de enumeratieklasse, en instanties van deze klasse voor elk van de geënumereerde waardes. De geënumereerde waardes worden ook met een **owl:oneOf** constructie begrensd door de enumeratieklasse. De SHACL gegevensregel maakt gebruik van het **sh:in** construct om de enumeratie uit te drukken.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:Golfbaan a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty imgolf:natuurwaarde ;
    owl:allValuesFrom imgolf:NatuurwaardeValue ;
  ] ;

.

imgolf:natuurwaarde a owl:ObjectProperty ;

imgolf:NatuurwaardeValue a owl:Class ;
  rdfs:subClassOf [
    a owl:Class ;
    owl:oneOf (imgolf:Hoog imgolf:Laag imgolf:Geen) ;
  ] ;

.

imgolf:Hoog a imgolf:NatuurwaardeValue .
imgolf:Laag a imgolf:NatuurwaardeValue .
imgolf:Geen a imgolf:NatuurwaardeValue .

#-----#
# Gegevensregels
#-----#

imgolf-sh:Golfbaan a sh:NodeShape ;
  sh:targetClass imgolf:Golfbaan ;
  sh:property [
    sh:path imgolf:natuurwaarde ;
    sh:in (imgolf:Hoog imgolf:Laag imgolf:Geen) ;
  ] ;

.
```

vergelijking met andere best practices

In de [Inspire RDF Guidelines](#) wordt voorgeschreven om een enumeratie te modelleren als **rdfs:Datatype** in plaats van als klasse [[INSPIRERDF](#)]. Dit leidt tot enumeratiewaarden die een literal zijn, met het datatype van de enumeratie.

Bijvoorbeeld "**hoog**"^^**imgolf:NatuurwaardeValue**. De reden om hiervan af te wijken is omdat enumeraties vaker waardelijsten zijn die een object of concept modelleren, dan een lijst van letterlijke waarden. Door deze waarden als objecten te modelleren blijft het mogelijk om nieuwe uitdrukkingen te doen over de waarden.

7.3.4.1.7 KLASSE MET STEREOTYPE «CODELIST»

Indien vooraf niet bekend is welke waarden een bepaald attribuut kan krijgen, maar er wel een lijst waarschijnlijke waarden is, dan wordt in plaats van een enumeratie een CodeList gebruikt. Een CodeList is een klasse met als stereotype «CodeList».

Gezien er structureel geen verschil is met een enumeratie, wordt een klasse met sterotype «CodeList» op dezelfde manier getransformeerd.

vergelijking met andere best practices

In de [Inspire RDF Guidelines](#) worden CodeLists gemodelleerd m.b.v. SKOS [[INSPIRERDF](#)]. Gezien het afhankelijk is van

de type waarden in de waardelijst, en de specifieke modelleringssstijl die wordt toegepast, of een modellering met SKOS toepasselijk is laten we deze keuze aan de handmatige transformatie over.

7.3.4.2 Attributen

7.3.4.2.1 ATTRIBUTEN ALGEMEEN

Een attribuut in een NEN 3610 UML model wordt vertaald naar een **owl:DatatypeProperty** wanneer het verwijst naar een letterlijk datatype. Hierbij wordt gebruik gemaakt van een owl:Restriction om de klasse waarop het attribuut voorkomt te koppelen en het bereik van de waarde te beschrijven.

In de bijbehorende gegevensregel wordt het attribuut aan de klasse gekoppeld via **sh:property** en **sh:path** en wordt het datatype beschreven met **sh:datatype**.

NOOT

We gebruiken geen **rdfs:domain** en **rdfs:range** in de standaard transformatie, omdat de semantische betekenis van **rdfs:domain** en **rdfs:range** niet altijd overeenkomt met de semantische betekenis van het UML attribuut.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:par a owl:DatatypeProperty .

imgolf:Parcours a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty imgolf:par ;
    owl:allValuesFrom xsd:integer ;
  ] ;

.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Parcours a sh:NodeShape ;
  sh:targetClass imgolf:Parcours ;
  sh:property [
    sh:path imgolf:par ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
.
```

NOOT

Wanneer een attribuut van een UML-klasse geen letterlijke waarde als bereik heeft, wordt deze behandeld als een associatie (zie § 7.3.4.3 [Associaties](#)). Dit is bijvoorbeeld het geval bij attributen die verwijzen naar klassen met stereotype «**dataType**» of «**enumeration**».

7.3.4.2.2 ATTRIBUUT MET STEREOTYPE «IDENTIFICATIE»

Er is geen transformatie nodig voor attributen met dit stereotype. Voorkomens van deze identificaties kunnen direct gebruik maken van de identificatie constructen in het [NEN3610 vocabulaire](#). Op basis van het NEN 3610 vocabulaire kan een identificatie-instantie er zo uit zien:

voorbeeld RDF representatie

```
ex:NL.IMGOLF.1-v01 a imgolf:Golfbaan ;
    nen3610:identificatie ex:id-NL.IMGOLF.1-v01 ;
    .

    ex:id-NL.IMGOLF.1-v01 a nen3610:Identificatie ;
        nen3610:namespace "NL.IMGOLF" ;
        nen3610:lokaalID "1" ;
        nen3610:versie "v01" ;           # optioneel
        rdf:value "NL.IMGOLF.1-v01" # optioneel om de samengestelde identificatie op te nemen
    .
```

7.3.4.2.3 ATTRIBUUT MET STEREOTYPE «VOIDABLE»

In [\[NEN3610\]](#) wordt het gebruik van het stereotype «voidable» voorgeschreven wanneer een attribuut van een object geen waarde heeft in de dataset, maar het object in de werkelijkheid wel een waarde voor het attribuut kan hebben.

In Linked Data heeft het niet opnemen van de triple die de waarde uitdrukt exact dezelfde betekenis als «voidable». Zoals beschreven in [hoofdstuk 4](#) is een principe van Linked Data de open-wereld-aanname. Dat betekent dat je er niet vanuit kunt gaan dat als iets niet is gezegd, het niet waar is. Je kunt immers niet weten of de uitdrukking niet ergens in de wereld wel gedaan is, of misschien nog gedaan gaat worden.

vergelijking met andere best practices

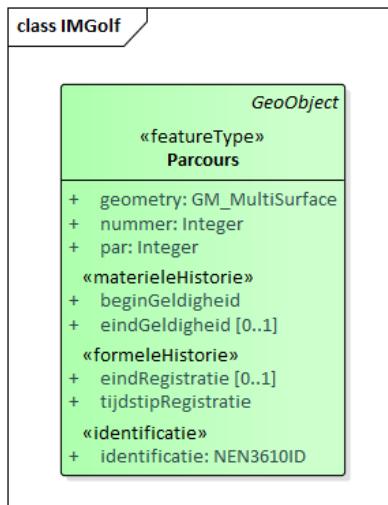
In de [Inspire RDF Guidelines](#) worden, om dezelfde redenen, voidable attributen zonder waarde niet opgenomen in de resulterende data [\[INSPIRERDF\]](#).

7.3.4.2.4 ATTRIBUUT MET STEREOTYPE «MATERIELEHISTORIE»

Een attribuut met stereotype «materieleHistorie» drukt de historie van veranderingen van eigenschappen van een object in de werkelijkheid uit [\[NEN3610\]](#). In een Linked Data context kunnen we dit scherper formuleren als: "een attribuut voor historie van geldigheid van een set uitdrukkingen over een object in de werkelijkheid". De historie die hiermee wordt beschreven slaat immers op de geldigheid van het geheel van eigenschappen van een object op een bepaald moment in de tijd.

Het uitdrukken van eigenschappen die over een set uitdrukkingen gaan is in RDF niet triviaal. Het is namelijk niet mogelijk om een set van uitdrukkingen in de subjectpositie van een triple te gebruiken. We hebben hier een manier nodig om een set bij elkaar horende triples te kunnen beschrijven als een ding.

Neem bijvoorbeeld de IMGolf klasse Parcours. Deze heeft de attributen [beginGeldigheid](#) en [eindGeldigheid](#) met stereotype «materieleHistorie».



Figuur 15 Attribuut met stereotypes «materieleHistorie» en «formeleHistorie»

Deze attributen slaan niet alleen op het Parcours als ding in de werkelijkheid, maar op de beschrijving van het Parcours met de eigenschappen zoals **nummer**, **par** en **geometry**.

Hier komt het maken van onderscheid tussen de non-information resource (Parcours nummer 4) en de information resource (een (web)document met een beschrijving van Parcours nummer 4), zoals beschreven in [5.2](#) te pas. De SHACL gegevensregels behorende bij de NEN 3610 ontologie biedt hier een constructie voor. Daarin wordt gebruik gemaakt van het [FOAF vocabulaire](#) voor het modelleren van information resources, zoals dat ook in [[LINKED-DATA-EVOLVING-WEB](#)] wordt toegepast.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:Parcours a owl:Class ;
    rdfs:subClassOf nen3610:GeoObject ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty imgolf:nummer ;
        owl:allValuesFrom xsd:integer ;
    ] ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty imgolf:par ;
        owl:allValuesFrom xsd:integer ;
    ] ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty foaf:isPrimaryTopicOf ;
        owl:allValuesFrom foaf:Document ;
    ] ;

.

imgolf:nummer a owl:DatatypeProperty .

imgolf:par a owl:DatatypeProperty .

#-----#
# Gegevensregels
#-----#

imgolf-sh:Parcours a sh:NodeShape ;
    sh:targetClass imgolf:Parcours ;
    sh:property [
        sh:path imgolf:nummer ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path imgolf:par ;
        sh:datatype xsd:integer ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;

.

#-----#
# NEN3610 Gegevensregels
#-----#

shape:GeoObject a sh:NodeShape ;
    sh:property [
        sh:name "foaf:isPrimaryTopicOf";
        sh:path foaf:isPrimaryTopicOf ;
        sh:node shape:GeoObjectRegistratie ;
        rdfs:comment "Deze eigenschap verbindt een Geo-object met de verschillende
versies van de beschrijving van dit geo-object" ;
    ] ;
.
```

```

shape:GeoObjectRegistratie a sh:NodeShape ;
  rdfs:label "Shape voor GeoObjectRegistratie" ;
  sh:property [
    sh:name "nen3610:beginGeldigheid" ;
    sh:path nen3610:beginGeldigheid ;
    sh:datatype xsd:dateTime ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
  sh:property [
    sh:name "nen3610:eindGeldigheid" ;
    sh:path nen3610:eindGeldigheid ;
    sh:datatype xsd:dateTime ;
    sh:maxCount 1 ;
  ] ;
.

#-----#
# Voorbeelddata
#-----#

ex:parcours-NL.IMGOLF.4 a imgolf:Parcours ;                               # De non-information resource
  rdfs:label "Parcours 4" ;
  nen3610:identificatie ex:id-NL.IMGOLF.4 ;
  imgolf:nummer 4 ;
  imgolf:par 3 ;
  foaf:isPrimaryTopicOf ex-doc:20180101/parcours-NL.IMGOLF.4; # verbinding van NIR met IR

.

ex-doc:20180101/parcours-NL.IMGOLF.4 a foaf:Document ;                      # De information resource
  rdfs:label "Beschrijving van Parcours 4" ;
  nen3610:beginGeldigheid "2018-01-01T00:00:00.000+01:00"^^xsd:dateTime ;
  nen3610:eindGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;
.

```

Voor het modelleren van een information resource gebruiken we de **foaf:Document** klasse. Voor het verbinden van de non-information resource met één of meerdere information resources gebruiken we de **foaf:isPrimaryTopicOf** eigenschap die aanduidt dat de non-information resource het hoofdonderwerp is van de information resource. Hiermee kunnen we op een standaard manier contextuele metadata, zoals materiële historie uitdrukken.

NOOT

Let op! Wanneer je in één dataset meerdere materieel geldige beschrijvingen van hetzelfde object in de werkelijkheid hebt is het noodzakelijk deze van elkaar te scheiden. Gezien de identificatie/IRI van het ding in de werkelijkheid niet veranderd, heb je een manier nodig om de bij elkaar horende set uitdrukkingen te scheiden van andere sets uitdrukkingen. Hiervoor kun je gebruik maken van (named) graphs.

vergelijking met andere best practices

Geen.

7.3.4.2.5 ATTRIBUUT MET STEREOTYPE «FORMELEHISTORIE»

Een attribuut met stereotype «formeleHistorie» drukt de historie van veranderingen van eigenschappen van een object in een registratie uit [NEN3610]. In een Linked Data context is het duidelijker om te zeggen: "Een attribuut met stereotype «formeleHistorie» is een attribuut voor historie van geldigheid van de registratie van een set uitdrukkingen over een object in de werkelijkheid".

Gezien formele historie, net als materiële historie, iets zegt over een set uitdrukkingen, kan dit op vergelijkbare manier gemodelleerd worden. Wanneer we uitgaan van hetzelfde voorbeeld als in [7.3.4.2.4](#) kunnen we formele historie als volgt modelleren.

voorbeeld RDF representatie

```

#-----#
# NEN3610 Gegevensregels
#-----#


shape:GeoObject a sh:NodeShape ;
  sh:property [
    sh:name "foaf:isPrimaryTopicOf";
    sh:path foaf:isPrimaryTopicOf ;
    sh:node shape:GeoObjectRegistratie ;
    rdfs:comment "Deze eigenschap verbindt een Geo-object met de verschillende
versies van de beschrijving van dit geo-object" ;
  ] ;

.

shape:GeoObjectRegistratie a sh:NodeShape ;
  rdfs:label "Shape voor GeoObjectRegistratie" ;
  sh:property [
    rdfs:label "tijdstip registratie" ;
    sh:name "prov:GeneratedAtTime" ;
    sh:path prov:generatedAtTime ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ];
  sh:property [
    rdfs:label "eindregistratie" ;
    sh:name "prov:InvalidatedAtTime" ;
    sh:path prov:invalidatedAtTime ;
    sh:maxCount 1 ;
  ];
  sh:property [
    rdfs:label "Voorbeelddata
#-----#
ex:parcours-NL.IMGOLF.4 a imgolf:Parcours ;                                # De non-information resource
  rdfs:label "Parcours 4" ;
  nen3610:identificatie ex:id-NL.IMGOLF.4 ;
  imgolf:nummer 4 ;
  imgolf:par 3 ;
  foaf:isPrimaryTopicOf ex-doc:20180101/parcours-NL.IMGOLF.4; # verbinding van NIR met IR
.

ex-doc:20180101/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;          # De information reso
  rdfs:label "Beschrijving van Parcours 4" ;

# materiele historie
nen3610:beginGeldigheid "2018-01-01T00:00:00.000+01:00"^^xsd:dateTime ;
nen3610:eindGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;

# formele historie
prov:generatedAtTime "2018-01-02T00:00:00.000+01:00"^^xsd:dateTime ;
prov:invalidatedAtTime "2019-04-19T00:00:00.000+02:00"^^xsd:dateTime ;
.

```

Door ook hier de set uitdrukkingen te representeren als information resource kunnen we uitdrukkingen doen over de formele historie van deze information resource. Dit maakt het mogelijk om verschillende information resources over dezelfde non-information resource naast elkaar te kunnen laten bestaan. Bij Linked Data publicatie kan hiermee de gewenste versie gepubliceerd worden. De NEN 3610 ontologie biedt ook hier een standaard shape constructie voor. Hier wordt gebruik gemaakt van de [PROV ontologie](#) om de registratiemetadata uit te drukken.

vergelijking met andere best practices

Geen.

7.3.4.2.6 ATTRIBUUT MET STEREOTYPE «MATERIELELEVENDUUR»

Een attribuut met stereotype «materieleLevensduur» drukt het complete tijdsinterval van geldigheid van het object in de werkelijkheid uit [[NEN3610](#)]. In een Linked Data context is het duidelijker om te zeggen: "Een attribuut met stereotype «materieleLevensduur» is een attribuut voor het vastleggen van de tijdsinterval van werkelijk voorkomen van geldige uitdrukkingen over een object in de werkelijkheid".

Net als materiële historie, zegt materiële levensduur iets over een set uitdrukkingen over een (geo-)object. Echter, het is in een linked data context niet correct om de eigenschap **beginTijd** uit [[NEN3610](#)] op het niveau van de information resource te plaatsen, gezien het daar geen intrinsieke eigenschap van is. Immers, dezelfde waarde voor de eigenschap zal dan worden herhaald op elke versie van de gegevensset over het object. Om deze reden is in de NEN 3610 ontologie een nieuwe eigenschap geïntroduceerd: **nen3610:vorigeGeldigeVersie**. Met deze eigenschap kan een opeenvolging van versies van materieel geldende beschrijvingen van een object uitgedrukt worden. Hiermee kan, in combinatie met de eigenschappen voor materiële historie, bepaald worden wat de materiële levensduur van een object is.

voorbeeld RDF representatie

```

#-----#
# NEN3610 Gegevensregels
#-----#

shape:GeoObject a sh:NodeShape ;
  sh:property [
    sh:name "foaf:isPrimaryTopicOf";
    sh:path foaf:isPrimaryTopicOf ;
    sh:node shape:GeoObjectRegistratie ;
    rdfs:comment "Deze eigenschap verbindt een Geo-object met de verschillende
    versies van de beschrijving van dit geo-object" ;
  ] ;

.

shape:GeoObjectRegistratie a sh:NodeShape ;
  rdfs:label "Shape voor GeoObjectRegistratie" ;
  sh:property [
    rdfs:label "heeft vorige materieel geldige versie" ;
    sh:name "nen3610:vorigeGeldigeVersie" ;
    sh:path nen3610:vorigeGeldigeVersie ;
    sh:maxCount 1 ;
  ] ;

.

#-----#
# Voorbeelddata
#-----#

ex-doc:20190322/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;
  rdfs:label "Beschrijving van Parcours 4 (2019-03-22 - heden)" ;

  # materiële historie
  nen3610:beginGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;

  # vorige materieel geldige versie
  nen3610:vorigeGeldigeVersie ex-doc:20180101/parcours-NL.IMGOLF.4 ;

.

ex-doc:20180101/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;
  rdfs:label "Beschrijving van Parcours 4 (2018-01-01 - 2019-03-22)" ;

  # materiële historie
  nen3610:beginGeldigheid "2018-01-01T00:00:00.000+01:00"^^xsd:dateTime ;
  nen3610:eindGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;
.
```

vergelijking met andere best practices

Geen.

7.3.4.2.7 ATTRIBUUT MET STEREOTYPE «FORMELELEVENDUUR»

Een attribuut met stereotype «formeleLevensduur» drukt de tijdsinterval van geldigheid van het object in de registratie uit [NEN3610]. In een Linked Data context is het duidelijker om te zeggen: "Een attribuut met stereotype «materieleLevensduur» is een attribuut voor het vastleggen van de tijdsinterval van het in een registratie voorkomen van uitdrukkingen over een object in de werkelijkheid".

Formele levensduur kan op dezelfde wijze als materiële levensduur gemodelleerd worden. Bij het modelleren van formele levensduur gebruiken we de eigenschap **nen3610:vorigeGeregistreerdeVersie** om de vorige formeel geregistreerde versie van een set gegevens over een object uit te drukken.

voorbeeld RDF representatie

```

#-----#
# NEN3610 Gegevensregels
#-----#

shape:GeoObject a sh:NodeShape ;
  sh:property [
    sh:name "foaf:isPrimaryTopicOf";
    sh:path foaf:isPrimaryTopicOf ;
    sh:node shape:GeoObjectRegistratie ;
    rdfs:comment "Deze eigenschap verbindt een Geo-object met de verschillende
      versies van de beschrijving van dit geo-object" ;
  ] ;

.

shape:GeoObjectRegistratie a sh:NodeShape ;
  rdfs:label "Shape voor GeoObjectRegistratie" ;
  sh:property [
    rdfs:label "heeft vorige formeel geldige versie" ;
    sh:name "nen3610:vorigeGeregistreerdeVersie" ;
    sh:path nen3610:vorigeGeregistreerdeVersie ;
    sh:maxCount 1 ;
  ] ;

.

#-----#
# Voorbeelddata
#-----#


ex-doc:201903221/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;
  rdfs:label "Beschrijving van Parcours 4 (2019-03-22 - heden) v1" ;

  # materiële historie
  nen3610:beginGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;

  # formele historie
  prov:generatedAtTime "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;

  # vorige formeel geldige versie
  nen3610:vorigeGeregistreerdeVersie ex-doc:201801012/parcours-NL.IMGOLF.4 ;

.

ex-doc:201801012/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;
  rdfs:label "Beschrijving van Parcours 4 (2018-01-01 - 2019-03-22) v2" ;

  # materiële historie
  nen3610:beginGeldigheid "2018-01-01T00:00:00.000+01:00"^^xsd:dateTime ;
  nen3610:eindGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;

  # formele historie
  prov:generatedAtTime "2019-04-19T00:00:00.000+02:00"^^xsd:dateTime ;

  # vorige formeel geldige versie
  nen3610:vorigeGeregistreerdeVersie ex-doc:201801011/parcours-NL.IMGOLF.4 ;

.

ex-doc:201801011/parcours-NL.IMGOLF.4 a foaf:Document, prov:Entity ;
  rdfs:label "Beschrijving van Parcours 4 (2018-01-01 - 2019-03-22) v1" ;

  # materiële historie
  nen3610:beginGeldigheid "2018-01-01T00:00:00.000+01:00"^^xsd:dateTime ;

```

```

nen3610:eindGeldigheid "2019-03-22T00:00:00.000+01:00"^^xsd:dateTime ;
# formele historie
prov:generatedAtTime "2018-01-02T00:00:00.000+01:00"^^xsd:dateTime ;
prov:invalidatedAtTime "2019-04-19T00:00:00.000+02:00"^^xsd:dateTime ;
.
```

vergelijking met andere best practices

Geen.

7.3.4.3 Associations

Associaties, of relaties, zijn verbanden die gelegd worden tussen klassen in een UML model.

7.3.4.3.1 SIMPELE ASSOCIATIES

Een simpele associatie is een associatie tussen twee klassen, zonder extra eigenschappen. Een simpele associatie in een NEN 3610 UML model wordt gemodelleerd als **owl:ObjectProperty** in een RDF model en wordt, op vergelijkbaar met attributen, gekoppeld met de klasse waarop het voorkomt en de klasse waarnaar de associatie verwijst.

De SHACL shape is ook vergelijkbaar met die van attributen. Het verschil hier is dat er in plaats van **sh:datatype**, **sh:class** wordt gebruikt voor het aanduiden van de klasse waarnaar de associatie verwijst.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#
imgolf:hole a owl:ObjectProperty .

imgolf:Golfbaan a owl:Class ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty imgolf:hole ;
        owl:allValuesFrom imgolf:Parcours ;
    ] ;
.

#-----#
# Gegevensregels
#-----#
imgolf-sh:Golfbaan a sh:NodeShape ;
    sh:targetClass imgolf:Golfbaan ;
    sh:property [
        sh:path imgolf:hole ;
        sh:class imgolf:Parcours ;
    ] ;
.
```

7.3.4.3.2 AGGREGATIE EN COMPOSITIE

Aggregatie- en compositie-associaties worden gemodelleerd als [simple associaties](#), gebruikmakend van de specifieke naam die de associatie in het oorspronkelijke model heeft.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:parcoursDeel a owl:ObjectProperty .

imgolf:ParcoursDeel a owl:Class .

imgolf:Parcours a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty imgolf:parcoursDeel ;
    owl:allValuesFrom imgolf:ParcoursDeel ;
  ] ;
.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Parcours a sh:NodeShape ;
  sh:targetClass imgolf:ParcoursDeel ;
  sh:property [
    sh:path imgolf:parcoursDeel ;
    sh:class imgolf:ParcoursDeel ;
    sh:minCount 1 ;
  ] ;
.
```

7.3.4.3.3 GENERALISATIE/SPECIALISATIE

Generalisatie- en specialisatie-associaties tussen klassen in een UML NEN 3610 model worden in RDF modellen gemodelleerd als subtype-associatie, uitgedrukt met een **rdfs:subClassOf** relatie tussen de betrokken klassen. Omdat SHACL node shapes van een klasse ook gelden voor haar subklassen wordt er enkel een node shape gemaakt wanneer de subklasse eigenschappen heeft die de superklasse niet heeft.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:Parcours a owl:Class ;
  rdfs:subClassOf nen3610:GeoObject ;
.
```

7.3.4.3.4 ASSOCIATIEKLASSEN

Een associatieklasse drukt een verband uit tussen twee objecten, waarbij er ook gegevens over dit verband worden vastgelegd. Bij de standaardtransformatie naar een RDF Linked Data modellering worden associatieklassen als normale klassen behandeld.

7.3.4.4 Kardinaliteit

Kardinaliteit in een NEN 3610 UML model geven aan hoeveel waardes een attribuut kan hebben, of hoe vaak een associatie op een instantie van een klasse kan voorkomen. RDF Linked Data gaat echter uit van de open world assumption. In OWL kun je wel kardinaliteit van eigenschappen uitdrukken om bepaalde (sub)sets van een werkelijkheid te beschrijven, maar niet om een aantal voorkomens van een eigenschap te beperken. Gezien het lastig is om, zonder uitleg van de maker van een UML model, te bepalen wat de betekenis is van de gemodelleerde kardinaliteiten worden er bij de standaard transformatie in de ontologie geen uitspraken over gedaan.

Met SHACL kunnen we wel de gesloten wereld beschrijven en kunnen we de kardinaliteit die het UML model uitdrukt prima aan. In SHACL gebruiken we daarvoor `sh:minCount` en `sh:maxCount` voor.

7.3.4.5 Overig

7.3.4.5.1 PACKAGE

In sommige UML modellen wordt gebruik gemaakt van een package om modelementen te groeperen. Soms kan de groepering gezien worden als één informatiemodel, maar dat is niet altijd het geval.

In de standaardtransformatie worden alle packages getransformeerd naar instanties van `owl:Ontology`. Alle modelementen die onderdeel zijn van deze package krijgen een relatie `rdfs:isDefinedBy` naar de instantie van de bijbehorende insantie van `owl:Ontology`.

NOOT

Het is aan te raden om handmatig te controleren of de opdeling in packages leidt tot de juiste ontologieën.

voorbeeld RDF representatie

```
#-----
# Ontologie
#-----

<http://definities.geostandaarden.nl/def/imgolf> a owl:Ontology ;
    rdfs:label "IMGolf" ;
    rdfs:comment "Package: IMGolf" ;

.

imgolf:Parcours a owl:Class ;
    rdfs:label "Parcours" ;
    rdfs:isDefinedBy <http://definities.geostandaarden.nl/def/imgolf> ;

.

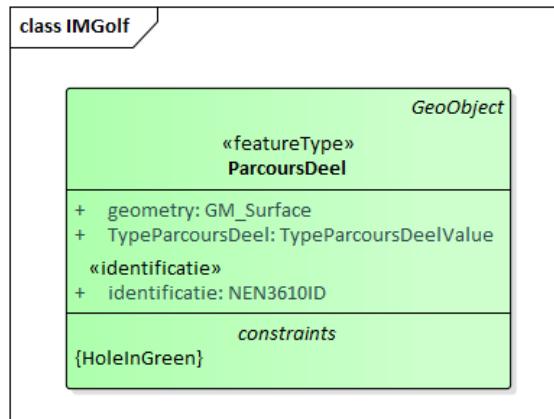
imgolf:natuurwaarde a owl:DatatypeProperty ;
    rdfs:label "natuurwaarde" ;
    rdfs:isDefinedBy <http://definities.geostandaarden.nl/def/imgolf> ;

.
```

7.3.4.5.2 CONSTRAINT

Een constraint is een beperking die één of meerdere modelementen betreft. Een constraint wordt vaak in natuurlijke taal omschreven en is daarom niet te vertalen naar een element in het doelmetamodel.

Bij de transformatie wordt de tekst van de constraint als extra toelichting bij de betreffende modelelementen opgenomen in de ontologie in de vorm van een [rdfs:comment](#).



Figuur 16 Klasse met constraint

In bovenstaand voorbeeld is er voor de klasse `ParcoursDeel` een constraint `{HoleInGreen}` gedefinieerd die aangeeft dat een hole zich binnen de geometrie van het parcoursdeel van het type `green` moet bevinden. Dit leidt tot de volgende transformatie naar de ontologie.

```
 voorbeeld RDF representatie

#-----#
# Ontologie
#-----#
imgolf:ParcoursDeel a owl:Class ;
    rdfs:label "ParcoursDeel" ;
    rdfs:subClassOf nen3610:GeoObject ;
    rdfs:comment """{HoleInGreen}:
/*hole bevindt zich binnen de geometrie van de green */
Inside(hole.geometrie,this->geometrie)""";

```

7.4 Transformatie: specifieke aanpassing - encoding

7.4.1 Inleiding

Zoals in [hoofdstuk 5](#) is uitgelegd, is het niet mogelijk om altijd een goed Linked Data model te verkrijgen uit een automatische vertaling van een UML-informatiemodel. Daarnaast zijn er in de praktijk verschillende metamodellen die gehanteerd worden bij het modelleren van Linked Data.

In dit hoofdstuk wordt voor drie metamodellen die in Nederland worden toegepast uiteengezet welke overwegingen er spelen bij het transformeren naar een RDF Linked Data model en welke specifieke transformatieregels er voor dat metamodel gehanteerd worden.

7.4.2 Omgevingswet-DSO: specifieke transformatieregels

7.4.2.1 Metamodel DSO

Het DSO hanteert het basis doelmodel gebaseerd op W3C standaarden, zoals beschreven in § 7.3.2 Basis doelmodel: [metamodel op basis van W3C standaarden](#) en afgebeeld in Figuur 9. Daarnaast wordt er gestreefd naar een genormaliseerd informatiemodel, om de herbruikbaarheid en linkbaarheid van de Linked Data te maximaliseren.

7.4.2.2 Transformatieregels DSO

7.4.2.2.1 KLASSEN

7.4.2.2.1.1 KLASSEN ALGEMEEN

Ten opzichte van de transformatieregels beschreven in § 7.3.4.1.1 [Klassen algemeen](#) zijn er enkele afwijkingen in de transformatie conform het DSO metamodel alsook een aantal toevoegingen.

AFWIJKINGEN

Daar waar een waarde voor `rdfs:label` gegeneerd is op basis van de klassenaam, wordt de waarde aangepast naar een label in een correct Nederlands.

voorbeeld RDF representatie

```
imgolf:ParcoursDeel a owl:Class ;
    #-- rdfs:label "ParcoursDeel" ; -- vervalt
    rdfs:label "Parcoursdeel" ;
    .
```

TOEVOEGINGEN

Het DSO metamodel is gebaseerd op het [basis doelmodel](#) en maakt onderscheid tussen terminologie (RDF, RDFS, OWL), formele semantiek (RDFS, OWL), gegevensregels (SHACL) en betekenis in de vorm van begrippen (SKOS). Voor een klasse wordt altijd, in ieder geval, een instantie van `owl:Class` en een `sh:NodeShape` gemaakt conform de standaard transformatieregels, maar ook een instantie van `skos:Concept` om een begrip te beschrijven die de betekenis van de klasse uitdrukt. Deze wordt vanuit de instantie van `owl:Class` verbonden via de `dct:subject` property.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:Parcours a owl:Class ;
    rdfs:label "Parcours" ;
    rdfs:comment """Een Parcours is een golfbaandeel dat van een tee via de fairway
(indien aanwezig) tot de bijbehorende green loopt""";
    dct:subject imgolf-beg:Parcours ;

.

#-----#
# Begrippen
#-----#

imgolf-beg:Parcours a skos:Concept ;
    skos:prefLabel "Parcours"@nl ;
    skos:definition """Een Parcours is een golfbaandeel dat van een tee via de fairway
(indien aanwezig) tot de bijbehorende green loopt""">@nl ;
    ...
        # verder beschrijving van het begrip conform SKOS
.

```

NOOT

Het definiëren van een **skos:Concept** wordt voor alle instanties van owl:Class gedaan.

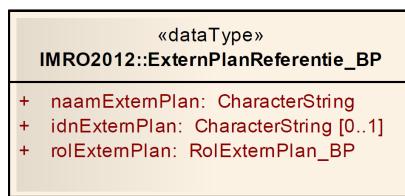
7.4.2.2.1.2 KLASSE MET STEREOTYPE «FEATURETYPE»

Geen afwijking, anders dan beschreven in § 7.4.2.2.1.1 [Klassen algemeen](#)

7.4.2.2.1.3 KLASSE MET STEREOTYPE «DATATYPE»

Zoals opgemerkt in § 7.3.4.1.3 [Klasse met stereotype «dataType»](#) is het bij het transformeren van een NEN 3610 UML klasse met stereotype «dataType» belangrijk om te goed te kijken naar wat het samenstel van attributen van de klasse uitdrukt.

Neem bijvoorbeeld dit datatype uit IMRO2012 die een relatie van het ene plan naar een ander plan beschrijft.



Figuur 17 Klasse met stereotype «dataType»

Het is uiteraard mogelijk om hier een klasse "ExternPlanReferentie" van te creëren, maar wat levert het op? Als we goed kijken naar wat hier wordt uitgedrukt, dan wordt hier een relatie gelegd naar een ander ding. Een van de principes die het DSO metamodel hanteert is dat een RDF Linked Data model genormaliseerd moet zijn. Dat betekent dat we dingen zo specifiek mogelijk onderscheiden, zodat derden vanuit hun eigen context (her)gebruik kunnen maken van de data die in het

model is uitgedrukt. In dit geval is er sprake van een verwijzing naar een plan én de rol die dat plan speelt ten op zichte van een ander plan. De rol kan in de Linked Data getransformeerd worden tot een property, die een instantie van een plan als bereik heeft. Zo kan worden uitgedrukt dat een instantie een specifieke relatie (o.b.v. de rol) heeft met een specifiek plan.

voorbeeld RDF representatie

```
ex:specifiekeRol a owl:ObjectProperty ;
  rdfs:subPropertyOf ex:planrelatie ; # Optioneel
  rdfs:range ex:Plan ;
  .

ex:Plan a owl:Class .
```

Hiermee hebben we het model opener en linkbaarder gemaakt. Stel dat er ergens meer informatie bestaat over hetzelfde plan, dan kan er met dit model nu een link gelegd worden naar de instantie van dat plan.

Een ander voorbeeld is de datatype-kLASSE **ToegangGolfbaan** uit het IMGOLF model. Een golfbaan heeft een attribuut **toegang** met als waardetype **ToegangGolfbaan**. **ToegangGolfbaan** heeft zelf de attributen **vereisten**, die een tekstuele beschrijving van de toegangseisen tot de golfbaan beschrijven, de waarde **etiquette**, die de etiquette van de golfbaan beschrijft en een eigen **toegang** attribuut die aangeeft of toegang tot de golfbaan openbaar of besloten is.

Wederom is dit wel een goed model om data over golfbanen in uit te drukken, maar is het geen goed genormaliseerd model van de werkelijkheid. De klasse **ToegangGolfbaan** groepeert wel een aantal eigenschappen die iets met de toegang tot golfbane te maken hebben, maar dat maakt niet dat er dingen, of concepten zijn in de werkelijkheid die deze eigenschappen hebben. Deze groepering maken we in het RDF Linked Data model dan ook niet. We modelleren deze eigenschappen als eigenschap van de golfbaan. **vereisten** wordt eigenschap **imgolf:toegangsvereisten**, **etiquette** wordt **imgolf:etiquette** en de **toegang** eigenschap van **ToegangGolfbaan** wordt **imgolf:toegankelijkheid**. Daarnaast verdwijnt de eigenschap **toegang** op de klasse **Golfbaan**.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:Golfbaan a owl:Class ;
  rdfs:subClassOf
  [
    a owl:Restriction ;
    owl:onProperty imgolf:toegangsvereisten ;
    owl:minCardinality 0 ;
  ] ,
  [
    a owl:Restriction ;
    owl:onProperty imgolf:etiquette ;
    owl:minCardinality 0 ;
  ] ,
  [
    a owl:Restriction ;
    owl:onProperty imgolf:toegankelijkheid ;
    owl:minCardinality 0 ;           # Kardinaliteit 0..* omdat we ontologisch
                                    # niet strict willen zijn
  ] ;

.

imgolf:toegangsvereisten a owl:DatatypeProperty ;
  rdfs:range xsd:string ;

.

imgolf:etiquette a owl:DatatypeProperty ;
  rdfs:range xsd:string ;

.

imgolf:toegankelijkheid a owl:ObjectProperty ;
  rdfs:range skos:Concept ;

.

#-----#
# Gegevensregels
#-----#


imgolf-sh:Golfbaan a sh:NodeShape ;
  sh:targetClass imgolf:Golfbaan ;
  sh:property
  [
    sh:path imgolf:toegangsvereisten ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
  ] ,
  [
    sh:path imgolf:etiquette ;
    sh:maxCount 1 ;
    sh:datatype xsd:string ;
  ] ,
  [
    sh:path imgolf:toegankelijkheid ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:node [
      sh:property [
        sh:path [ sh:inversePath skos:member ] ; # onderdeel van skos:Collection
      ]
    ]
  ]
}

```

```

sh:hasValue imgolf-collectie:Toegankelijkheid ;
] ;
] ;
]
```

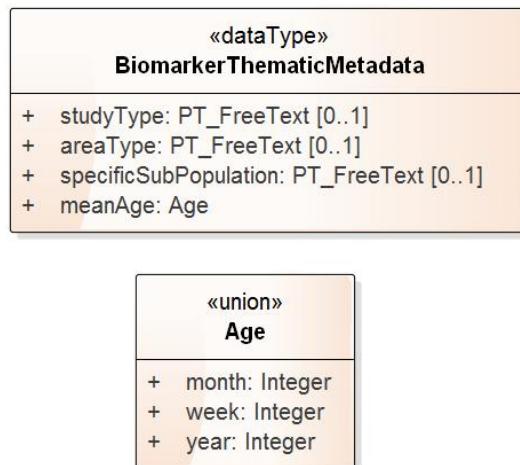
NOOT

Merk op dat deze voorbeelden geen uitputtende lijst aan mogelijke transformaties van klassen met stereotype «*dataType*» zijn. Het is aan de modelleur, of transformeerder, om te bepalen wat de beste vertaling is naar een Linked Data model, rekening houdend met de principes van Linked Data.

7.4.2.2.1.4 KLASSE MET STEREOTYPE «UNION»

Voor unions van dingen wordt dezelfde modellering als de standaard transformatie gehanteerd. Wanneer de union niet een keuze tussen verschillende dingen represeneert, wordt gekeken naar de beste manier om de specifieke klasse te modelleren.

Neem het eerdere voorbeeld van een union van literal waarden uit de [Inspire RDF Guidelines \[INSPIRERDF\]](#).



*Figuur 18 Klasse met stereotype «union» - een union van literal waarden
bron: [INSPIRERDF]*

In dit geval is het semantisch correcter om af te wijken van de standaardtransformatie. De modellering die in het Inspire voorbeeld wordt uitgevoerd is een betere en semantisch correctere modellering. In onderstaande uitwerking beschrijven we als aanvulling op dat [voorbeeld](#) de SHACL `sh:NodeShape` die er mee correspondeert. Onze shape bevat een `sh:xone` conditie die uitdrukt dat exact één van de properties `:meanAgeByMonth`, `:meanAgeByWeek` of `:meanAgeByYear` aanwezig moet zijn.

voorbeeld RDF representatie

```
#-----#
# Gegevensregels
#-----#

bio-sh:BiomarkerThematicMetadata a sh:NodeShape ;
    sh:targetClass :BiomarkerThematicMetadata ;
    sh:xone (
        [
            sh:property [
                sh:path :meanAgeByMonth ;
                sh:datatype xsd:integer ;
            ] ;
        ]
        [
            sh:property [
                sh:path :meanAgeByWeek ;
                sh:datatype xsd:integer ;
            ] ;
        ]
        [
            sh:property [
                sh:path :meanAgeByYear ;
                sh:datatype xsd:integer ;
            ] ;
        ]
    ) ;
.
```

NOOT

Net als bij andere stereotypen is het niet mogelijk om een uitputtende lijst aan transformatieregels te geven. Er is te veel variatie mogelijk in bron UML modellen om tot eenduidige transformatieregels te komen die tot een goed RDF Linked Data model komen.

7.4.2.2.1.5 KLASSE MET STEREOTYPE «EXTERNAL»

In de standaard transformatieregels wordt een klasse met stereotype «external» herkenbaar getransformeerd door de een van te voren afgesproken afwijkende namespace te gebruiken voor de IRI van de klasse. In de specifieke DSO-transformatieregels worden deze klassen waar mogelijk vervangen door, of gerelateerd aan, bestaande ontologieën. Wanneer deze ontologieën niet bestaan wordt de klasse in eigen context gemodelleerd.

Wanneer we het voorbeeld uit IMGOLF nemen waar een golfbaan een relatie **BAGAdres** heeft met een klasse met stereotype «external» **AdresserbaarObject**, kunnen we als volgt direct de IRI van de bestaande klasse **bag:AdresseerbaarObject** uit de BAG-ontologie gebruiken.

voorbeeld RDF representatie

```
## Deze klasse vervalt
# external:AdresserbaarObject a owl:Class .

imgolf:Golfbaan a owl:Class ;
owl:subClassOf [
  a owl:Restriction ;
  owl:onProperty imgolf:bagAdres ;
  owl:allValuesFrom bag:AdresseerbaarObject ; # gebruikt direct externe IRI.
] ;

.

imgolf:bagAdres a owl:ObjectProperty ;
rdfs:range bag:AdresseerbaarObject .           # gebruikt direct externe IRI.
```

7.4.2.2.1.6 KLASSE MET STEREOTYPE «ENUMERATION»

Zoals opgemerkt in § 7.3.4.1.6 [Klasse met stereotype «enumeration»](#) kan een enumeratie verschillende soorten dingen opsommen. Bij verschillende dingen horen verschillende aanpakken, immers, we willen dingen zo specifiek mogelijk modelleren, zodat anderen ze kunnen herkennen en (her)gebruiken of eraan linken.

ENUMERATIES VAN TYPES

Zoals beschreven in § 5.6 [Setoriëntatie in Linked Data en de relatie met de UML Class](#) is het in RDF gewoon om dingen te klassificeren tot meerdere klassen. Wanneer er sprake is van een typerende enumeratie (meestal zijn dit lijsten die de 'soort' of 'type' in de naam hebben), wordt dit ook als klasse gemodelleerd. Wel is het zaak om goed te kijken of de typerende lijst als subklassen van de bestaande klasse gemodelleerd moet worden, of als losstaande klassehiërarchie.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:Golfbaan a owl:Class .

imgolf:Linksbaan a owl:Class ;
    rdfs:subClassOf imgolf:Golfbaan ;

.

imgolf:Bosbaan a owl:Class ;
    rdfs:subClassOf imgolf:Golfbaan ;

.

imgolf:Parkbaan a owl:Class ;
    rdfs:subClassOf imgolf:Golfbaan ;

.

imgolf:Parcoursdeel a owl:Class .

imgolf:Fairway a owl:Class ;
    rdfs:subClassOf imgolf:Parcoursdeel ;

.

imgolf:Green a owl:Class ;
    rdfs:subClassOf imgolf:Parcoursdeel ;

.

imgolf:Rough a owl:Class ;
    rdfs:subClassOf imgolf:Parcoursdeel ;

.

imgolf:SemiRough a owl:Class ;
    rdfs:subClassOf imgolf:Parcoursdeel ;

.

imgolf:Hindernis a owl:Class .

imgolf:Waterpartij a owl:Class ;
    rdfs:subClassOf imgolf:Hindernis ;

.

imgolf:Bunker a owl:Class ;
    rdfs:subClassOf imgolf:Hindernis ;

.

```

NOOT

Het is ook mogelijk om een typerende enumeratie van relatiesoorten tegen te komen. In dit geval wordt hetzelfde patroon toegepast om specifieke attributen of associaties te modelleren, gebruikmakende van **rdfs:subPropertyOf** in plaats van **rdfs:subClassOf** om hiërarchie daarbij aan te brengen.

ENUMERATIES VAN CONCRETE DINGEN

Wanneer een enumeratie bestaat uit dingen uit een werkelijkheid is het van belang om dit ook als een set dingen te modelleren. De rationale is dat het mogelijk is dat dezelfde dingen elders ook beschreven zijn. In dat geval is het mogelijk om de dingen semantisch aan elkaar te koppelen en kunnen we de kracht van linked data benutten om onze gegevens te verrijken. In linked data betekent dit het modelleren als instanties van een specifieke klasse.

Een voorbeeld van een dergelijke enumeratie is een enumeratie van landen. Een land is een concreet iets in de institutionele werkelijkheid. Het is ook iets waar vanuit verschillende bronnen informatie over is vastgelegd. Daarom is het van belang om een land als ding te modelleren.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#
ex:Land a owl:Class .  
  
#-----#
# Gegevens
#-----#  
  
land:528 a ex:Land ;
    rdfs:label "Nederland"@nl ;
    ex:iso-3166-1-alpha-2-code "NL" ;
    owl:sameAs ;  
  
.  
  
land:276 a ex:Land ;
    rdfs:label "Duitsland"@nl ;
    ex:iso-3166-1-alpha-2-code "DE" ;
    owl:sameAs ;  
  
.  
...  
...
```

ENUMERATIES VAN CONCEPTUELE DINGEN

Wanneer er sprake is van een enumeratie van conceptuele, of abstracte dingen, wordt er in de DSO aanpak gebruik gemaakt van de gemodelleerde SKOS concepten die die dingen representeren. Omdat het hier niet nodig is om deze dingen verder te beschrijven worden ze niet in OWL uitgewerkt. In vrijwel alle gevallen gaat het hier om categoriserende enumeraties, zoals ook het geval in onderstaand voorbeeld.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#
imgolf:Golfbaan a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty imgolf:natuurwaarde ;
    owl:qualifiedCardinality 1 ;
    owl:onClass [
      owl:oneOf (
        concept:HogeNatuurwaarde
        concept:LageNatuurwaarde
        concept:GeenNatuurwaarde
      ) ;
    ] ;
  ] ;

.

imgolf:natuurwaarde a owl:ObjectProperty ;
  rdfs:domain imgolf:Golfbaan ;
  rdfs:range skos:Concept ;

.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Golfbaan a sh:NodeShape ;
  sh:targetClass imgolf:Golfbaan ;
  sh:property [
    sh:path imgolf:natuurwaarde ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:node imgolf-sh:NatuurwaardeValue ;
  ] ;

.

imgolf-sh:NatuurwaardeValue a sh:NodeShape ;
  sh:property [
    sh:path [ sh:inversePath skos:member] ;
    sh:hasValue collectie:NatuurwaardeValue ;
  ] ;

.

#-----#
# Begrippen
#-----#

concept:Natuurwaarde a skos:Concept ;
  skos:inScheme schema:IMGolf ;
  skos:topConceptOf schema:IMGolf ;
  skos:prefLabel "natuurwaarde" ;
  skos:narrower
    concept:HogeNatuurwaarde ,
    concept:LageNatuurwaarde ,
    concept:GeenNatuurwaarde ;

.

concept:HogeNatuurwaarde a skos:Concept ;
  skos:inScheme schema:IMGolf ;

```

```

skos:prefLabel "hoge natuurwaarde" ;
skos:notation "hoog" ;
skos:broader concept:Natuurwaarde ;

concept:LageNatuurwaarde a skos:Concept ;
skos:inScheme schema:IMGolf ;
skos:prefLabel "lage natuurwaarde" ;
skos:notation "laag" ;
skos:broader concept:Natuurwaarde ;

concept:GeenNatuurwaarde a skos:Concept ;
skos:inScheme schema:IMGolf ;
skos:prefLabel "geen natuurwaarde" ;
skos:notation "geen" ;
skos:broader concept:Natuurwaarde ;

collectie:NatuurwaardeValue a skos:Collection ;
rdfs:label "Waardelijst NatuurwaardeValue" ;
skos:inScheme schema:IMGolf ;
skos:member
concept:HogeNatuurwaarde ,
concept:LageNatuurwaarde ,
concept:GeenNatuurwaarde ;

```

ENUMERATIES VAN LETTERLIJKE WAARDEN

Als laatste onderscheiden we een enumeratie van letterlijke waarden. Hierbij moet goed gekeken worden of de enumeratie niet voldoet aan een van de hierboven beschreven enumeraties. Een voorbeeld is een lijst van toegestane namen, of een lijst van toegestane nummers. Het is afhankelijk van de aard van het lijstje wat de beste modelleringswijze is. Voor sommige opsommingen kan de standaard transformatie voor enumeraties volstaan. Voor anderen is het wellicht beter om het uit te drukken in een SHACL constraint die de lading dekt.

7.4.2.2.1.7 KLASSE MET STEREOTYPE «CODELIST»

Zoals eerder opgemerkt wordt een klasse met stereotype «CodeList» op dezelfde manier behandeld als een klass stereotype «enumeration». Voor illustratieloeinden wordt hieronder alsnog een voorbeeld van de klasse met stereotype «CodeList» **GebruiksstatusValue** uit het IMGOLF uitgewerkt.

Omdat de codelijst abstracte concepten modelleert, maken we gebruik van instanties van **skos:Concept** om deze codelijst uit te drukken. Voor elk van de codes wordt een concept aangemaakt en voor elke code-waarde uit de oorspronkelijke codelijst krijgt het betreffende concept een eigenschap **skos:notation** met de exacte waarde uit de oorspronkelijke lijst. Hiermee kunnen dezelfde enumeratiewaarden uit de oorspronkelijke lijst gebruikt worden vanuit de Linked Data.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:Golfbaan a owl:Class .

imgolf:gebruiksstatus a owl:ObjectProperty ;
  rdfs:range skos:Concept .

.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Golfbaan a sh:NodeShape ;
  sh:targetClass imgolf:Golfbaan ;
  sh:property [
    sh:path imgolf:gebruiksstatus ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
    sh:node [
      sh:property [
        sh:path [ sh:inversePath skos:member ] ;
        sh:hasValue imgolf-collectie:GebruiksstatusValue ;
      ] ;
    ] ;
  ] ;
.

#-----#
# Begrippen
#-----#

imgolf-beg:InUitvoering a skos:Concept ;
  skos:prefLabel "In uitvoering"@nl ;
  skos:broader imgolf-beg:Gebruiksstatus ;
  skos:notation "in uitvoering" ;           # code-waarde

.

imgolf-beg:InGebruik a skos:Concept ;
  skos:prefLabel "In gebruik"@nl ;
  skos:broader imgolf-beg:Gebruiksstatus ;
  skos:notation "in gebruik" ;             # code-waarde

.

imgolf-beg:BuitenGebruik a skos:Concept ;
  skos:prefLabel "Buiten gebruik"@nl ;
  skos:broader imgolf-beg:Gebruiksstatus ;
  skos:notation "buiten gebruik" ;         # code-waarde

.

imgolf-collectie:GebruiksstatusValue a skos:Collection ;
  skos:member
    imgolf-beg:InUitvoering ,
    imgolf-beg:InGebruik ,
    imgolf-beg:BuitenGebruik .
.

```

7.4.2.2.2 ATTRIBUTEN

7.4.2.2.2.1 ATTRIBUTEN ALGEMEEN

Ten opzichte van de transformatieregels beschreven in § 7.3.4.2.1 [Attributen algemeen](#) zijn er enkele afwijkingen in de transformatie conform het DSO metamodel alsook een aantal toevoegingen.

AFWIJKINGEN

Daar waar een waarde voor `rdfs:label` gegenereerd is op basis van de eigenschapnaam, wordt de waarde aangepast naar een label in correct Nederlands.

Daar waar het semantisch klopt kan in plaats van, of naast de standaard gemodelleerde instanties van `owl:Restriction` gebruik gemaakt worden van `rdfs:domain` en/of `rdfs:range`, om respectievelijk uit te drukken dat de eigenschap altijd voorkomt op instantie van een bepaalde klasse, of altijd verwijst naar een instantie van een bepaald klasse. Met name met het toepassen van `rdfs:domain` moet secuur gebeuren, om semantische onjuistheden te voorkomen.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:Parcours a owl:Class ;
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:onProperty imgolf:nummer ;
        owl:someValuesFrom xsd:integer ;
    ] ;
    .

imgolf:nummer a owl:DatatypeProperty ;
    rdfs:range xsd:integer ;
    .
```

In bovenstaand voorbeeld kiezen we ervoor om alleen de `rdfs:range` te specificeren op het attribuut `imgolf:nummer`, omdat een nummer typisch iets is wat aan verschillende typen dingen toegewezen kan worden.

TOEVOEGINGEN

Voor een eigenschap wordt altijd ook een instantie van `skos:Concept` gemodelleerd om een begrip te beschrijven die de betekenis van het concept achter de eigenschap uitdrukt. Deze wordt vanuit de instantie van de property verbonden via de `dct:subject` property.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:par a owl:DatatypeProperty ;
    rdfs:label "par" ;
    rdfs:comment "Een Parcours is een golfbaandeel dat van een tee via de fairway
(indien aanwezig) tot de bijbehorende green loopt" ;
    dct:subject imgolf:beg:Par ;
    rdfs:domain imgolf:Parcours ;
    rdfs:range xsd:string ;

.

#-----#
# Begrippen
#-----#

imgolf:beg:Par a skos:Concept ;
    rdfs:label "Par" ;
    skos:prefLabel "Par" ;
    skos:altLabel "Par staat voor Professional Average Result." ;
    skos:inScheme schema:IMGolf ;
    skos:definition "het aantal slagen waarin een gemiddelde professionele golfer
een hole zou moeten kunnen spelen." ;
    skos:related concept:Parcours ;
    dct:source ;
.
```

7.4.2.2.2 ATTRIBUUT MET STEREOTYPE «IDENTIFICATIE»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.2.2.3 ATTRIBUUT MET STEREOTYPE «VOIDABLE»

Wanneer het echt noodzakelijk is om informatie over het ontbreken van voidable informatie vast te leggen is het mogelijk om dit te koppelen aan de beschrijving (de information resource) van een object. Dit kan op vergelijkbare manier als bij § 7.3.4.2.4 [Attribuut met stereotype «materieleHistorie»](#). Het is in dat geval aan te raden om hier een specifieke klasse voor te definiëren.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

nen3610:voided a owl:AnnotationProperty ;
    rdfs:domain foaf:Document ;
    rdfs:range void:VoidValue ;

.

nen3160:VoidValue a owl:Class .

nen3610:voidedProperty a owl:ObjectProperty ;
    rdfs:domain nen3160:VoidValue ;
    rdfs:range rdf:Property ;

.

nen3610:voidReason a owl:ObjectProperty ;
    rdfs:domain nen3160:VoidValue ;
    rdfs:range xsd:string ;

.

#-----#
# Gegevensregels
#-----#

nen3610-sh:GeoObject a sh:NodeShape ;
    sh:property [
        sh:path foaf:isPrimaryTopicOf ;
        sh:node nen3610-sh:GeoObjectRegistratie ;
        rdfs:comment "Deze eigenschap verbindt een Geo-object met de verschillende
versies van de beschrijving van dit geo-object" ;
    ] ;

.

nen3610-sh:GeoObjectRegistratie a sh:NodeShape ;
    sh:property [
        sh:path nen3610:voided ;
        sh:node nen3610-sh:VoidValue ;
    ] ;

.

nen3610-sh:VoidValue a sh:NodeShape ;
    sh:property [
        sh:path nen3610:voidedProperty ;
        sh:class rdf:Property ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path nen3610:voidReason ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
.
```

7.4.2.2.4 ATTRIBUUT MET STEREOTYPE «MATERIELEHISTORIE»

Geen afwijking ten op zichte van de standaard transformatieregels.

7.4.2.2.5 ATTRIBUUT MET STEREOTYPE «FORMELEHISTORIE»

Geen afwijking ten op zichte van de standaard transformatieregels.

7.4.2.2.6 ATTRIBUUT MET STEREOTYPE «MATERIELELEVENSDUUR»

Geen afwijking ten op zichte van de standaard transformatieregels.

7.4.2.2.7 ATTRIBUUT MET STEREOTYPE «FORMELELEVENSDUUR»

Geen afwijking ten op zichte van de standaard transformatieregels.

7.4.2.2.3 ASSOCIATIES

Associaties, of relaties, zijn verbanden die gelegd worden tussen klassen in een UML model.

7.4.2.2.3.1 SIMPELE ASSOCIATIES

Geen afwijking ten opzichte van de standaard transformatieregels, anders dan beschreven bij § 7.4.2.2.2 [Attributen](#)

7.4.2.2.3.2 AGGREGATIE EN COMPOSITIE

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.2.2.3.3 GENERALISATIE/SPECIALISATIE

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.2.2.3.4 ASSOCIATIEKLASSEN

In UML bestaat de notie van associatieklassen, ook wel relatieklassen genaamd. Een associatieklasse is een associatie met eigenschappen. Omdat RDF uitdrukkinggeoriënteerd is kent het geen associatieklassen. Er zijn verschillende manieren om in RDF het doel wat een associatieklasse dient te bereiken. Eén manier is de relatie modelleren als object, zoals in de standaardtransformatie wordt gedaan. Dit heeft als voordeel dat de uitdrukkingen die hieruit volgen simpel te bevragen zijn met SPARQL en gemakkelijker te mappen zijn op een objectgeoriënteerd paradigma. Het nadeel is dat het context-specifieke complexiteit introduceert in een informatiemodel. Dit maakt dat het informatiemodel, en de informatie daarin uitgedrukt, minder gemakkelijk her te gebruiken is in een andere context.

De RDFS recommendation [[rdf-schema](#)] biedt daarnaast de [Reification Vocabulary](#). Dit vocabulaire introduceert een manier om een specifieke uitdrukking (triple) te "verdingen", zodat het gebruikt kan worden als subject in nieuwe uitdrukkingen. Het voordeel hiervan is dat het informatiemodellen contextloos houdt. In feite wordt het hiermee mogelijk om contextuele informatie over een uitdrukking met een apart stuk model te beschrijven. Het nadeel hiervan is dat het moeilijker te bevragen is in SPARQL en dat het tot onbeslisbare modellen in OWL kan leiden. Desalniettemin is dit voor de DSO de voorkeursmethode voor het beschrijven van eigenschappen over een specifieke uitdrukking.

NOOT

Wanneer het noodzakelijk is om over een groepje uitdrukkingen eigenschappen vast te leggen kan gebruik gemaakt worden van [foaf:Document](#) reification, zoals in § 7.3.4.2.4 [Attribuut met stereotype «materieleHistorie»](#)

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

ex:beginDatum a owl:DatatypeProperty ;
    rdfs:range xsd:date ;
.

ex:eindDatum a owl:DatatypeProperty ;
    rdfs:range xsd:date ;
.

ex:gehuwdMet a owl:ObjectProperty ;
    rdfs:domain ex:Persoon ;
    rdfs:range ex:Persoon ;

.

ex:Huwelijk a owl:Class ;
    rdfs:subClassOf rdf:Statement ;
.

#-----#
# Gegevensregels
#-----#

ex-sh:Huwelijk a sh:NodeShape ;
    sh:targetClass ex:Huwelijk ;
    sh:property
    [
        [
            sh:path rdf:subject ;
            sh:class ex:Persoon ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
        ] ,
        [
            sh:path rdf:predicate ;
            sh:hasValue ex:gehuwdMet ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
        ] ,
        [
            sh:path rdf:object ;
            sh:class ex:Persoon ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
        ] ,
        [
            sh:path ex:beginDatum ;
            sh:datatype xsd:date ;
            sh:minCount 1 ;
            sh:maxCount 1 ;
        ] ,
        [
            sh:path ex:eindDatum ;
            sh:datatype xsd:date ;
            sh:maxCount 1 ;
        ] ;
.

#-----#
```

```
# Gegevens
#-----
:persoon1 a ex:Persoon ;
  ex:gehuwdMet :persoon2 ;
.

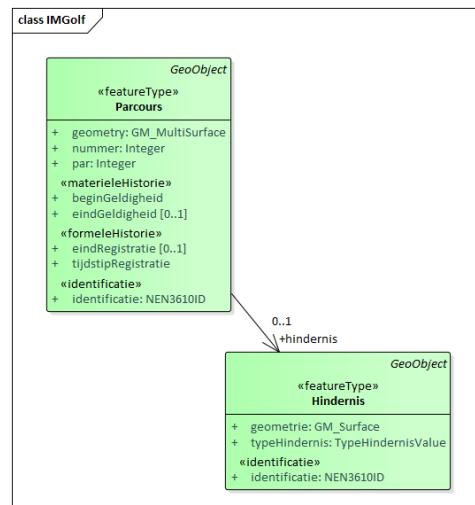
:huwelijk-p1-p2 a ex:Huwelijk ;
  rdf:subject :persoon1 ;
  rdf:predicate ex:gehuwdMet ;
  rdf:object :persoon2 ;
  ex:beginDatum "2018-04-20"^^xsd:date ;
.
```

7.4.2.2.4 KARDINALITEIT

Zoals al opgemerkt in betekent kardinaliteit in OWL iets anders dan kardinaliteit in UML. De "open-wereld" kardinaliteit in OWL is existentiell, in de zin dat de kardinaliteit die beschreven is over een eigenschap in relatie tot een klasse, in alle gevallen (binnen en buiten een registratie) zo is. In UML beschrijft de kardinaliteit hoe een gegevensobject in een specifieke situatie moet zijn, bijv. in een registratie, of in een servicebericht. Dit zegt niets over de existentiële eigenschappen van een object in de werkelijkheid en beschrijft een "gesloten wereld". Kardinaliteit beschreven in SHACL komt exact overeen met de betekenis in UML.

Bij de DSO wordt in ieder geval altijd de kardinaliteit in SHACL beschreven. Waar gewenst en mogelijk wordt de existentiële kardinaliteit met OWL beschreven.

In het IMGolf model is gespecificeerd dat Parcours een hindernisrelatie heeft met kardinaliteit 0 of 1. Dit kan prima gelden voor de golfbanen die beschreven worden met IMGolf.



Figuur 19 Kardinaliteit van de hindernisrelatie tussen Parcours en Hindernis

Er bestaan echter ook golfbanen met parcourses met meer dan 1 hindernis, bijvoorbeeld met bunkers én een waterpartij. Dit geeft aan dat de kardinaliteit in het IMGolf een gegevensregel is voor een administratie van specifieke IMGolf golfbanen, geen ontologische/existentiële regel. Wanneer we de kardinaliteit van de relaties ook als kardinaliteit in OWL zouden beschrijven, zou omze IMGolf ontologie niet meer semantisch verenigbaar zijn met andere golfbaan-ontologieën en daarmee gegevens over ander golfbanen. In dit geval modelleren we deze kardinaliteit als specifieke gegevensregel met SHACL. In OWL specificeren we hier geen kardinaliteit, omdat er semantisch geen restrictie op deze relatie ligt.

voorbeeld RDF representatie

```
#-----#
# Ontologie
#-----#

imgolf:hindernis a owl:ObjectProperty ;
  rdfs:domain imgolf:Parcours ;
  rdfs:range imgolf:Hindernis ;

.

imgolf:Parcours a owl:Class .

#-----#
# Gegevensregels
#-----#
imgolf-sh:Parcours a sh:NodeShape ;
  sh:targetClass imgolf:Parcours ;
  sh:property [
    sh:path imgolf:hindernis ;
    sh:class ex:Hindernis ;
    sh:maxCount 1 ;
  ] ;
.
```

7.4.2.2.5 OVERIG

7.4.2.2.5.1 PACKAGE

Zoals aangegeven in § [7.3.4.5.1 Package](#) moet handmatig bekeken worden of de onderverdeling in ontologieën een logische is. Dit kan betekenen dat meerdere instanties van [owl:Ontology](#) gecombineerd moeten worden tot één, of dat er handmatig een instantie van [owl:Ontology](#) moet worden toegevoegd.

7.4.2.2.5.2 CONSTRAINT

Van elke constraint moet worden bekeken of deze om te zetten is naar modelconstructies die de constraint kunnen uitdrukken.

In het voorbeeld in § [7.3.4.5.2 Constraint](#) wordt voor de klasse [ParcoursDeel](#) een constraint [{HoleInGreen}](#) gedefinieerd die aangeeft dat een hole zich binnen de geometrie van het parcoursdeel van het type [green](#) moet bevinden. Omdat dit om een ruimtelijke relatie gaat en omdat de klasse [imgolf:ParcoursDeel](#) een sub-klasse is van [nen3610:GeoObject](#) die zelf weer een sub-klasse is van [gsp:Feature](#), kunnen we gebruikmaken van de ruimtelijke relaties gedefinieerd voor [gsp:Feature](#). In dit geval kunnen we aangeven dat de eigenschap [imgolf:hole](#) een sub-property van [gsp:sfContains](#) is. Hiermee geven we aan dat wanneer deze relatie gebruikt wordt tussen een green en een hole, het ook betekent dat de hole binnen de green valt. Deze relatie kunnen we vervolgens afdwingen met behulp van een SHACL property shape.

voorbeeld RDF representatie

```

#-----#
# Ontologie
#-----#

imgolf:ParcoursDeel a owl:Class ;
  rdfs:label "ParcoursDeel" ;
  rdfs:subClassOf nen3610:GeoObject ;
  rdfs:comment """{HoleInGreen}:
    /*hole bevindt zich binnen de geometrie van de green */
    Inside(hole.geometrie,this->geometrie)""";
.

imgolf:Green a owl:Class ;
  rdfs:label "Green" ;
  rdfs:subClassOf imgolf:ParcoursDeel ;
.

imgolf:hole a owl:ObjectProperty ;
  rdfs:subPropertyOf gsp:sfContains ;
  rdfs:domain imgolf:Green ;
  rdfs:range imgolf:Hole ;
.

#-----#
# Gegevensregels
#-----#

imgolf-sh:Green a sh:NodeShape ;
  sh:targetClass imgolf:Green ;
  sh:property [
    sh:message "hole bevindt zich binnen de geometrie van de green" ;
    sh:path gsp:sfContains ;
    sh:class imgolf:Hole ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
  sh:property [
    sh:message "green moet exact één hole hebben" ;
    sh:path imgolf:hole ;
    sh:class imgolf:Hole ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;
.
.
```

7.4.3 Bouwsector - Linked Data: Specifieke transformatieregels

In de bouwsector wordt Linked Data gebruikt bij het modelleren van zogenaamde 'Object Type Libraries' of OTL's. Bij het modelleren hiervan is lange tijd automatisch uitgegaan van het zogenaamde 'objectificeren' van eigenschappen en relaties. Het COINS metamodel (CBIM) biedt hier een raamwerk voor en wordt hieronder verder toegelicht.

Naast het COINS raamwerk is er ook een 'Nederlandse Technische Afspraak', de NTA8035, waarin richtlijnen zijn vastgelegd voor de toekomstvaste modellering en integratie (uitwisseling en/of deling) van digitale gegevens die betrekking hebben op 'assets' in de context van de gebouwde omgeving zoals gebouwen, wegen en bruggen.

7.4.3.1 Conceptueel Top Model NTA8035

De NTA8035 bestaat uit een subset van concepten en relaties uit de nog in ontwikkeling zijnde nieuwe versie van de NEN2660. Waarbij in de NTA gekozen is voor taalbinding naar SKOS/RDFS/OWL. Daarmee wordt er invulling gegeven aan het Conceptueel Metamodel van de NEN2660.

De afspraken in de NTA8035 worden in twee internationale standaardisatietrajecten ingebracht, namelijk de ISO 21597 “Information Container for Data Delivery (ICDD)” [[iso-21597-1](#)] en de CEN TC442/WG4/TG3 “Semantic Modelling & Linking Standard (SMLS) for life-cycle information in the built environment”[[CEN-TC-442](#)].

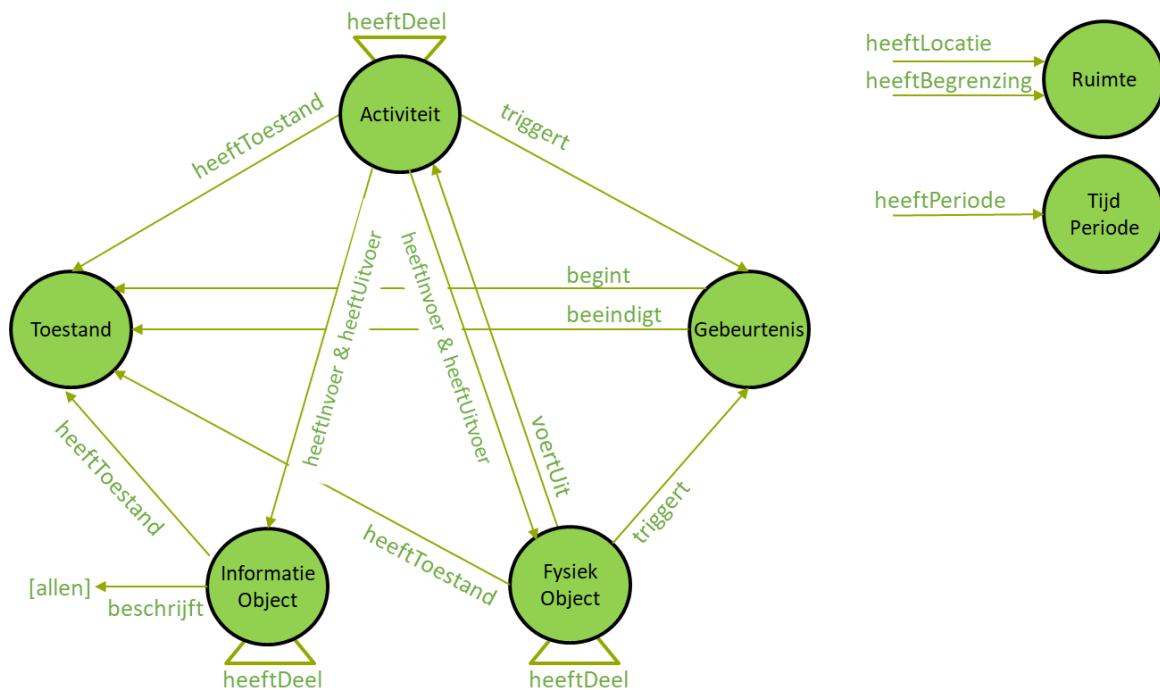
De NTA8035 definieert drie ambitieniveaus van toenemende semantische complexiteit:

1. Afstemming van termen en definities, gebaseerd op [SKOS](#)
(LoC-1)
2. Gegevensintegratie door uitwisseling of deling, gebaseerd op [RDFS](#)
(LoC-2)
3. Verificatie of Afleiding ([OWL](#) voor "open world" en [SHACL](#) voor "closed world" interpretatie) van gegevens uit bestaande gegevens
(Loc-3a en LoC-3b)

In zijn algemeenheid wordt er geadviseerd zo dicht mogelijk bij de standaard Linked Data constructies te blijven (de standaard transformatieregels in dit document). Voor complexe eigenschappen wordt een uitgebreide variant van de 'Ontology for Property Management' [[OPM](#)] gevuld. Grootheden en eenheden worden gemodelleerd via [[QUDT](#)].

Behalve modelleringsregels bevat de NTA 8035 ook een top model met een zevental concepten en een aantal relaties daartussen:

1. Fysiek object
2. Informatie object
3. Activiteit
4. Toestand
5. Gebeurtenis
6. Ruimte
7. Tijdperiode



(inverse relaties en details weggelaten voor beter overzicht)

Figuur 20 Top Level concepten, relaties en beperkingen schematisch gecombineerd

7.4.3.2 Transformatieregels NTA 8035

De § 7.3.4 Standaard transformatieregels die in dit document beschreven staan zijn gebaseerd op RDFS (wat overeenkomt met LoC-2 uit de NTA) en SHACL (wat overeenkomt met (LoC-3b uit de NTA). De onderstaande specifieke transformatieregels zijn van toepassing op de LoC-2 en LoC-3 niveau's uit de NTA.

7.4.3.2.1 KLASSEN

7.4.3.2.1.1 KLASSE MET STEREOTYPE «FEATURETYPE»

Klassen met stereotype featureType worden een subklasse van de betreffende topmodel klasse.

voorbeeld RDF representatie

```

nen3610:GeoObject
  rdfs:subClassOf bs:PhysicalObject .
  
```

7.4.3.2.1.2 KLASSE MET STEREOTYPE «DATATYPE»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.1.3 KLASSE MET STEREOTYPE «UNION»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.1.4 KLASSE MET STEREOTYPE «EXTERNAL»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.1.5 KLASSE MET STEREOTYPE «ENUMERATION»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.1.6 KLASSE MET STEREOTYPE «CODELIST»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.2 ATTRIBUTEN

Datatype properties worden in principe hetzelfde vertaald als in de standaard transformatie.

Daar waar voor het attribuut een grootheid of eenheid relevant is wordt de [QUDT 2.1 ontologie gevolgd](#).

Voor de attribuut stereotypen zijn in de NTA geen specifieke transformatieregels voorgeschreven.

7.4.3.2.2.1 ATTRIBUUT MET STEREOTYPE «IDENTIFICATIE»

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.2.2 ATTRIBUUT MET STEREOTYPE «VOIDABLE»

Niet beschreven in de NTA. In § [7.4.2.2.3 Attribuut met stereotype «voidable»](#) (DSO transformatieregels) wordt wel een manier beschreven om om te gaan met 'attributen waarvan je weet dat je ze niet weet'. Van belang om daarbij op te merken is dat er een onderscheid gemaakt wordt tussen de information resource en de non-information resource. Waarbij er dus specifiek vastgelegd wordt dat je de waarde van het attribuut niet weet in de betreffende registratie.

7.4.3.2.2.3 ATTRIBUUT MET STEREOTYPE «MATERIELEHISTORIE»

Niet beschreven in de NTA. In § [7.3.4.2.4 Attribuut met stereotype «materieleHistorie»](#) (standaard transformatieregels) wordt een manier beschreven om om te gaan met het vastleggen van materiële historie.

7.4.3.2.2.4 ATTRIBUUT MET STEREOTYPE «FORMELEHISTORIE»

Niet beschreven in de NTA. In § [7.3.4.2.5 Attribuut met stereotype «formeleHistorie»](#) (standaard transformatieregels) wordt een manier beschreven om om te gaan met het vastleggen van formele historie.

7.4.3.2.3 ASSOCIATIES

Associaties, of relaties, zijn verbanden die gelegd worden tussen klassen in een UML model.

7.4.3.2.3.1 SIMPELE ASSOCIATIES

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.3.2 AGGREGATIE EN COMPOSITIE

Voor alle associaties en specialisaties van compositie worden ook hun “inverse” gegeven.

RELATIE	INVERSE RELATIE
Associaties	
[InformatieObject] beschrijft [Allen]	[Allen] isBeschrevenDoor [InformatieObject]
[FysiekObject] heeftToestand [Toestand]	[Toestand] isToestandVan [FysiekObject]
[FysiekObject] triggert [Gebeurtenis]	[Gebeurtenis] isGetriggeredDoor [FysiekObject]
[FysiekObject] voertUit [Activiteit]	[Activiteit] isUitgevoerdDoor [FysiekObject]
[FysiekObject] heeftPeriode [TijdPeriode]	[TijdPeriode] isPeriodeVoor [FysiekObject]
[FysiekObject] heeftLocatie [Ruimte]	[Ruimte] isLocatieVan [FysiekObject]
[FysiekObject] heeftBegrenzing [Ruimte]	[Ruimte] isBegrenzingVan [FysiekObject]
[Activiteit] heeftToestand [Toestand]	[Toestand] isToestandVan [Activiteit]
[Activiteit] triggert [Gebeurtenis]	[Gebeurtenis] isGetriggeredDoor [Activiteit]
[Activiteit] heeftInvoer [FysiekObject]	[FysiekObject] isInvoerVan [Activiteit]
[Activiteit] heeftUitvoer [FysiekObject]	[FysiekObject] isUitvoerVan [Activiteit]
[Activiteit] heeftInvoer [InformatieObject]	[InformatieObject] isInvoerVan [Activiteit]
[Activiteit] heeftUitvoer [InformatieObject]	[InformatieObject] isUitvoerVan [Activiteit]
[Activiteit] heeftPeriode [TijdPeriode]	[TijdPeriode] isPeriodeVoor [Activiteit]
[Activiteit] heeftLocatie [Ruimte]	[Ruimte] isLocatieVan [Activiteit]
[Activiteit] heeftBegrenzing [Ruimte]	[Ruimte] isBegrenzingVan [Activiteit]
[Gebeurtenis] begint [Toestand]	[Toestand] isBegonnenDoor [Gebeurtenis]
[Gebeurtenis] beeindigt [Toestand]	[Toestand] isBeeindigdDoor [Gebeurtenis]
[State] heeftPeriode [TijdPeriode]	[TijdPeriode] isPeriodeVoor [Toestand]
[Toestand] heeftLocatie [Ruimte]	[Ruimte] isLocatieVan [Toestand]
[Toestand] heeftBegrenzing [Ruimte]	[Ruimte] isBegrenzingVan [Toestand]
Compositie/decompositie	
[FysiekObject] heeftDeel [FysiekObject]	[FysiekObject] isDeelVan [FysiekObject]
[InformatieObject] heeftDeel [InformatieObject]	[InformatieObject] isDeelVan [InformatieObject]
[Activiteit] heeftDeel [Activiteit]	[Activiteit] isDeelVan [Activiteit]

Figuur 21 Top Level relaties

Voor decompositie wordt de Dublin Core hasPart relatie gebruikt (en de inverse daarvan isPartOf)

voorbeeld RDF representatie

```



```

Daar waar de decompositie afhankelijk is van een specifieke view (Bijvoorbeeld SE of Geo) wordt aangeraden om verschillende ontologieën te maken.

7.4.3.2.3.3 GENERALISATIE/SPECIALISATIE

Geen afwijking ten opzichte van de standaard transformatieregels.

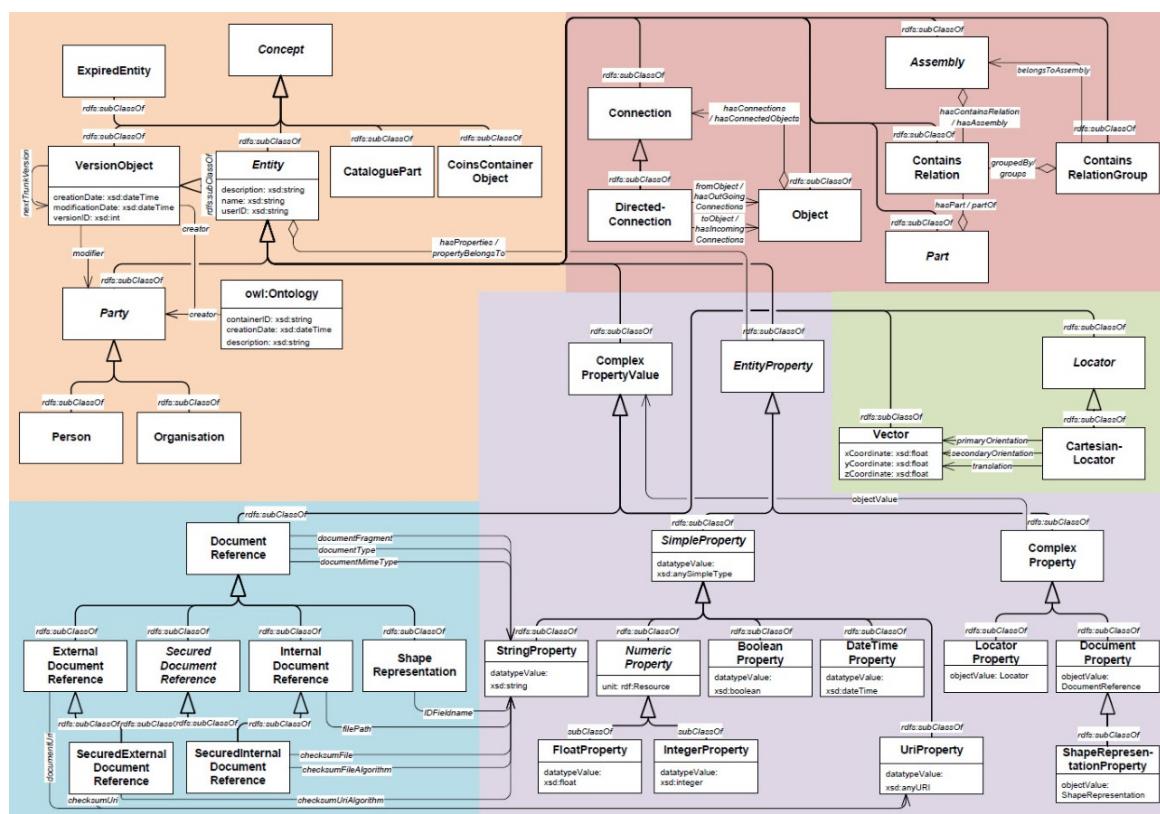
7.4.3.2.3.4 ASSOCIATIEKLASSEN

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.2.4 KARDINALITEIT

Geen afwijking ten opzichte van de standaard transformatieregels.

7.4.3.3 Metamodel COINS



Figuur 22 Metamodel van COINS

7.4.3.4 Transformatieregels COINS

COINS (Constructieve Objecten en de INtegratie van Processen en Systemen) is een flexibele standaard voor de uitwisseling van BIM-informatie. De standaard zorgt ervoor dat verschillende soorten informatie in samenhang in één

database kunnen worden vastgelegd, zoals functies, eisen- en objectenbomen, GIS-data, 2D-tekeningen, 3D-modellen, IFC-modellen, en objecttype-bibliotheek. De kern wordt gevormd door een neutraal, software-onafhankelijk datamodel, van waaruit data in een neutraal formaat kunnen worden verzonden en ‘vertaald’ naar de software van verschillende projectpartners.

In COINS2 is gekozen voor het ‘objectificeren’ van relaties en kenmerken waardoor deze expliciet refereerbaar worden en dus o.a. makkelijk te verrijken zijn met extra informatie zoals bijvoorbeeld meta-informatie.

Voor meer informatie zie: [coinsweb](#)

Het IMGolf model is geïmplementeerd als een objecttypebibliotheek die in een COINS container gebruikt kan worden. Hiervoor zijn de volgende stappen doorlopen vanuit het 'standaard' [RDF](#) model:

1. Importeren van het kernmodel van COINS2: <http://www.coinsweb.nl/cbim-2.0.rdf#>
2. Het NEN 3610 model zelf transformeren naar een [OTL](#)
3. Het IMGolf 'standaard' model transformeren naar een [OTL](#)

Het uitwisselen van Geometrie kan in COINS op verschillende manieren worden geïmplementeerd. Een mogelijkheid is om een ShapeRepresentation instantie op te nemen met een verwijzing naar bijvoorbeeld een [GML](#) file. Een andere mogelijkheid is om een geosparql geometrie op te nemen als eigenschap bij een COINS object. In het IMGolf voorbeeld is er voor gekozen om de geometrie op de geosparql wijze met geo:AsWKT op te nemen.

In het OTL-BOR project, uitgevoerd door provincies Noord-Holland en Gelderland, Gemeentes Rotterdam en Amsterdam en de kennisinstituten Geonovum en CROW in samenwerking met BuildingBits is er een proof-of-concept uitgevoerd met bovenstaande transformatieregels om van een DSO model naar een COINS model te komen. Meer informatie hierover kan gevonden worden in het [Eindrapport OTL-BOR](#).

7.4.3.4.1 KLASSEN

7.4.3.4.1.1 KLASSE MET STEREOTYPE «FEATURETYPE»

Klassen met stereotype featureType worden een subklasse van Object en van CataloguePart.

voorbeeld RDF representatie
<pre>nen3610:GeoObject rdfs:subClassOf cbim-2.0:CataloguePart ; rdfs:subClassOf cbim-2.0:Object ;</pre>

7.4.3.4.1.2 KLASSE MET STEREOTYPE «DATATYPE»

Klassen met stereotype dataType worden een subklasse van ComplexPropertyValue.

voorbeeld RDF representatie
<pre>IMGolf-otl:ToegangGolfbaan rdf:type owl:Class ; rdfs:subClassOf cbim-2.0:CataloguePart ; rdfs:subClassOf cbim-2.0:ComplexPropertyValue ;</pre>

7.4.3.4.1.3 KLASSE MET STEREOTYPE «UNION»

Voor een union gelden in principe geen andere regels dan de standaard transformatie.

7.4.3.4.1.4 KLASSE MET STEREOTYPE «EXTERNAL»

Voor een external type wordt een ObjectProperty voor aangemaakt met als range het externe object.

voorbeeld RDF representatie

```
IMGolf-otl:golfbaan_heeft_AdresseerbaarObject
  rdf:type owl:ObjectProperty ;
  rdfs:domain IMGolf-otl:Golfbaan ;
  rdfs:range <http://bag.basisregistraties.overheid.nl/def/bag#AdresseerbaarObject> ;
  rdfs:subPropertyOf cbim-2.0:hasProperties ;
```

7.4.3.4.1.5 KLASSE MET STEREOTYPE «ENUMERATION»

Waardelijsten worden vertaald naar coins:StringProperties

7.4.3.4.1.6 KLASSE MET STEREOTYPE «CODELIST»

Waardelijsten worden vertaald naar coins:StringProperties

voorbeeld RDF representatie

```
IMGolf-otl:Natuurwaarde
  rdf:type owl:Class ;
  rdfs:subClassOf cbim-2.0:CataloguePart ;
  rdfs:subClassOf cbim-2.0:StringProperty ;
  .

IMGolf-otl:geen
  rdf:type IMGolf-otl:Natuurwaarde ;
  .

IMGolf-otl:hoog
  rdf:type IMGolf-otl:Natuurwaarde ;
  .

IMGolf-otl:laag
  rdf:type IMGolf-otl:Natuurwaarde ;
  .
```

7.4.3.4.2 ATTRIBUTEN

Datatype properties worden vertaald naar COINS Kenmerken:

DatatypeProperties worden ObjectProperties als subproperties van coins:hasProperties

DatatypeProperties informatie wordt gebruikt om een subklasse van COINS:EntityProperty te creëren

Domain en ranges worden goed gezet

voorbeeld RDF representatie

```

IMGolf-otl:parcours_heeft_nummer
    rdf:type owl:ObjectProperty ;
    rdfs:domain IMGolf-otl:Parcours ;
    rdfs:range IMGolf-otl:Nummer ;
    rdfs:subPropertyOf cbim-2.0:hasProperties ;

.

IMGolf-otl:Nummer
    rdf:type owl:Class ;
    rdfs:subClassOf cbim-2.0:CataloguePart ;
    rdfs:subClassOf cbim-2.0:IntegerProperty ;
.
```

Voor de attribuut stereotypen zijn voor COINS geen specifieke transformatieregels voorgeschreven.

7.4.3.4.3 ASSOCIATIES

Associaties worden vertaald naar coins:Connections of coins:ContainsRelation met bijbehorende structuur. N.B. Dit is een stap die extra 'semantische duiding' toevoegt aan het model die in de BIM wereld veelal gewenst is. Deze transformatie is moeilijk in een generieke regel te definiëren.

voorbeeld RDF representatie

```

IMGolf-otl:GolfbaanHeeftHole
    rdf:type owl:Class ;
    rdfs:label "Golfbaan heeft hole (parcours)" ;
    rdfs:subClassOf IMGolf-otl:HeeftHole ;
    rdfs:subClassOf [
        rdf:type owl:Restriction ;
        owl:allValuesFrom IMGolf-otl:Golfbaan ;
        owl:onProperty cbim-2.0:hasAssembly ;
    ] ;
    rdfs:subClassOf [
        rdf:type owl:Restriction ;
        owl:allValuesFrom IMGolf-otl:Parcours ;
        owl:onProperty cbim-2.0:hasPart ;
    ] ;
.
```

7.4.4 Stedelijk Water - GWSW-OroX: Specifieke transformatieregels

7.4.4.1 Transformatieregels OroX: Maak een ontologie

Voor een goed begrip van de transformatieregels eerst een korte introductie van het GWSW-OroX.

Het GWSW-OroX protocol is ontwikkeld voor de beschrijving van het GegevensWoordenboek Stedelijk Water (GWSW). Oorspronkelijk was het GWSW (een ontwikkeling van Stichting RIONED) geschreven in de Gellish taal, een semantische rijke modelleringsstaal in het zogenaamde ORO (Object-Relatie-Object) formaat. Het GWSW onderscheidt zich daarmee door diepgang in semantiek en reikwijdte in de toepassing (van systeem tot proces).

Met de komst van het OroX protocol is het GWSW uitgedrukt in RDF, RDFS en OWL 2 (zie [GWSW Ontologie in RDF](#)

([pdf](#)). Het protocol beschrijft zowel het GWSW model (de "TBox", zie [data.gsws.nl](#)) als de daarop gebaseerde datasets (de "ABox", zie [apps.gsws.nl](#)).

NOOT

Modelleren van een ontologie met OroX

Wikipedia vermeldt: "een ontologie is typisch een datastructuur die alle relevante entiteiten en hun onderlinge relaties en regels binnen een domein bevat". Het GWSW-OroX protocol ondersteunt de beschrijving van zulke ontologien.

De datastructuur is object-georiënteerd waarbij relaties tussen objecten in een aantal structuren zijn ondergebracht:

1. Soortenboom (de taxonomie of klasse-indeling)
2. Samenstelling (de meronomie of deel-geheel indeling)
3. Proces (activiteiten-schema)

Belangrijke superklassen zijn gsws:FysiekObject, gsws:Activiteit, gsws:Ruimte, gsws:Levenvorm.

Bij het ontwerp van een datastructuur spelen deze elementen de hoofdrol, ze vormen het ontwerp kader.

Met het principe van object-oriëntatie hanteert het model overerving-principes en maakt het zo explicet mogelijk onderscheid in subklassen van de genoemde superklassen. Dat is een heel andere benadering dan bijvoorbeeld het ontwerp van een relationeel model. Daarbij ligt de nadruk ligt op het interpreteren van informatie met een hoofdrol voor de eerder besproken normalisatie-techniek om opslagruimte te beperken en redundantie te voorkomen.

Een voorbeeld van de object-georiënteerde benadering met onderscheid in fysiek object en activiteit:

Van elk parcours willen we het moment van grasmaaien kunnen achterhalen. We onderscheiden dan twee concepten.

Hole2	is een	Parcours
	heeft naam	"Hole 2"
Grasmaaien1	is een	ActiviteitGrasmaaien
	heeft datum uitvoering	"20190531"
	heeft uitvoerder	Grasmaaier1

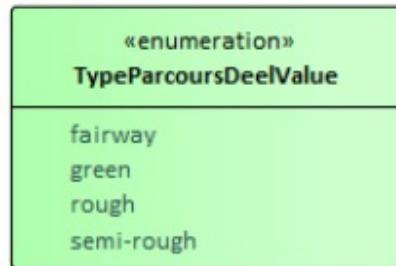
Vervolgens worden deze concepten als proces beschreven:

Grasmaaien1	heeft als invoer	Hole2
-------------	------------------	-------

Onderscheid de objecten

In de datastructuur worden alle entiteiten zo explicet mogelijk beschreven. Om die reden zullen objecttyperingen niet als domeinslijst opduiken.

De domeinwaardenlijst in het UML van NEN3610-IMGolf:



Figuur 23 Domeinwaarden type parcours

Wordt conform GWSW-OroX als volgt gemodelleerd:

voorbeeld RDF representatie
<pre> <imgolf:afslagplaatswit "2018-04-19"^^xsd:date="" "afslagplaats="" .="" .<="" ;="" <imgolf:afslagplaatsblauw="" <imgolf:afslagplaatsgeel="" <imgolf:afslagplaatsrood="" blauw"@nl="" geel"@nl="" gsws:hasdatestart="" imgolf:tee="" owl:class="" pre="" rdf:type="" rdfs:label="" rdfs:subclassof="" rood"@nl="" wit"@nl=""> </imgolf:afslagplaatswit></pre>

Onderscheidende kenmerken van objecten

De OWL semantiek wordt uitgebreid toegepast in het GWSW-OroX. OWL helpt om de taxonomie (de klasse-indeling, soortenboom) zo expliciet mogelijk te beschrijven. Daarmee kan "determinerend" de soort (of klasse) worden afgeleid uit de eigenschappen.

Bijvoorbeeld een "Parcours" heeft aan de start altijd een "Tee" (afslagplaats). Als in een dataset "iets" een "Tee" bevat kan de RDF-Reasoner afleiden dat het "iets" mogelijk een "Parcours" is.

De definitie van Parcours in RDF wordt dan:

voorbeeld RDF representatie

```

gsws:Parcours           rdfs:subClassOf
[ 
  rdf:type             owl:Restriction ;
  rdfs:label           "wsw:hasPart_Tee" ;
  owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
  owl:onProperty        gsws:hasPart ;
  owl:onClass            gsws:Tee
] .

```

Een "Afslagplaats geel" is een soort "Tee" (afslagplaats) en heeft als functie "Afslaan voor dummies". Met OWL 2 kan dat als volgt worden uitgedrukt:

De definitie van de "Afslagplaats geel":

voorbeeld RDF representatie

```

gsws:AfslagplaatsGeel
  rdf:type          owl:Class ;
  rdfs:label        "Afslagplaats geel"@nl ;
  rdfs:subClassOf   gsws:Tee ;

```

De definitie van de functie-instantie "Afslaan voor dummies":

voorbeeld RDF representatie

```

gsws:VoorDummies
  rdf:type          gsws:Functie ;
  rdfs:label        "Afslaan voor dummies"@nl .

```

Vervolgens "Afslagplaats geel" beschrijven als subklasse van de functie-restrictie

voorbeeld RDF representatie

```

gsws:AfslagplaatsGeel           rdfs:subClassOf
[ 
  rdf:type             owl:Restriction ;
  owl:onProperty       gsws:hasAspect ;
  owl:someValuesFrom [
    [
      owl:intersectionOf ( gsws:Functie
      [
        [
          rdf:type          owl:Restriction ;
          owl:onProperty     gsws:hasReference ;
          owl:hasValue       gsws:VoorDummies ] ) ]
  ] .

```

Afhankelijk van het gehanteerde OWL-expressieniveau kan een RDF-Reasoner uit onderstaande beschrijving afleiden dat bim:Iets mogelijk een "Afslagplaats geel" is:

voorbeeld RDF representatie

```

bim:Iets
  rdf:type          gsws:Tee ;
  gsws:hasAspect [
    [
      rdf:type gsws:Functie ;
      gsws:hasReference gsws:VoorDummies ;
    ] .

```

Collecties conform OroX

Vooral voor objecttypes is de expliciteit belangrijk. Ook het GWSW-OroX hanteert collecties, maar dan in de vorm van individuals (instanties van klassen). Bijvoorbeeld de waarden van kenmerken, die kunnen als individuals verschijnen.

De domeinwaardenlijst in het UML van NEN3610-IMGolf:



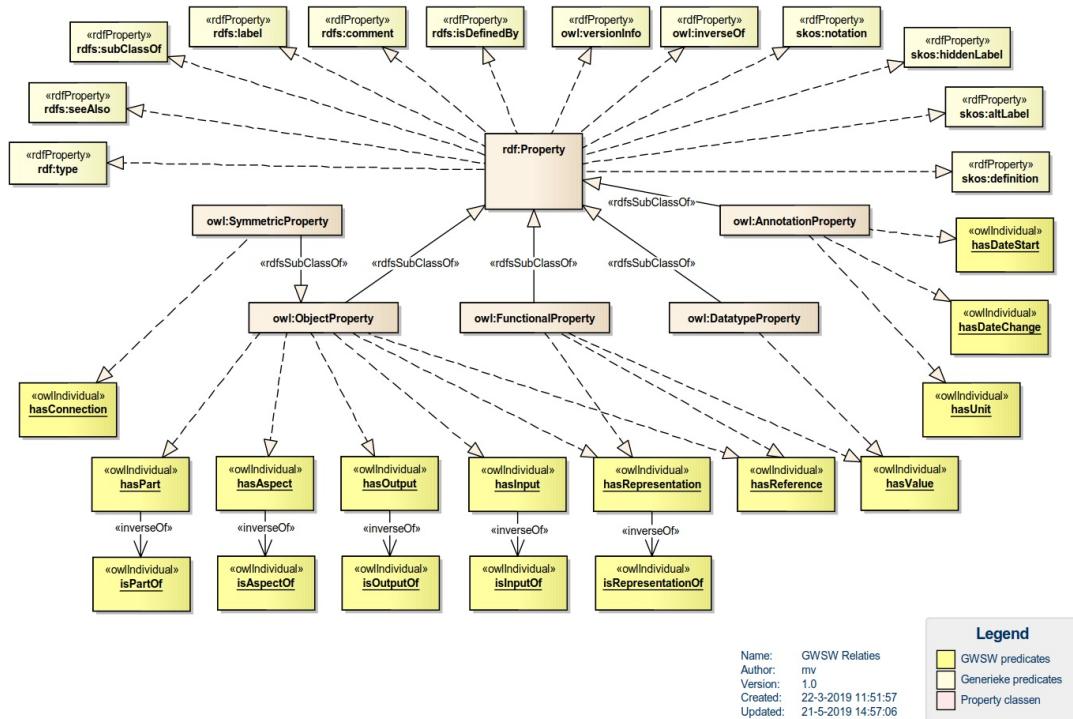
Figuur 24 Domeinwaarden natuurwaarde

Wordt in het GWSW-OroX als lijst van individuals gemodelleerd:

voorbeeld RDF representatie
<pre> @prefix imgolf: <http://www.w3.org/ns/govocab#> . @prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . imgolf:NatuurwaardeColl rdf:type owl:Class ; rdfs:label "Natuurwaarde (coll)"@nl ; gsws:Verzameling ; rdfs:subClassOf [rdf:type owl:Class ; owl:oneOf (imgolf:Hoog imgolf:Laag imgolf:Geen)] . imgolf:Hoog rdf:type imgolf:NatuurwaardeColl ; rdfs:label "Hoog"@nl . imgolf:Laag rdf:type imgolf:NatuurwaardeColl ; rdfs:label "Laag"@nl . imgolf:Geen rdf:type imgolf:NatuurwaardeColl ; rdfs:label "Geen"@nl . </pre>

7.4.4.2 Top-model GWSW-OroX

De oorspronkelijke Gellish-taal kent een groot aantal vooraf gedefinieerde relaties ("predicates"). In het GWSW-OroX zijn deze relaties sterk samengevat maar nog wel voorgescreven, het OroX kan gezien worden als een RDF-schema. In de volgende figuur staan schematisch de GWSW Relaties:



Figuur 25 Metamodel GWSW-OroX Relaties

De toepassing van predicates (per klasse) is in de GWSW-Ontologie vaak aan regels gebonden door middel van een Class Expression (CE). In de volgende tabel is dat aangegeven met “CE”.

Predicate	Inverse property	Type	Omschrijving
<code>owl:versionInfo</code>			<i>Subject (ontologie) heeft versieomschrijving Literal</i>
<code>rdf:type</code>			<i>Subject is van het type Object (Klasse-naam)</i>
<code>rdfs:subClassOf</code>			<i>Subject is van het subtype Object (Klasse-naam)</i>
<code>rdfs:label</code>			<i>Subject heeft als naam Literal (ook vertalingen, dan meerdere rdfs:label properties) (annotatie)</i>
<code>skos:altLabel</code>			<i>Subject heeft als synoniem Literal (ook vertalingen, dan meerdere rdfs:altLabel properties) (annotatie)</i>
<code>skos:hiddenLabel</code>			<i>Subject heeft als id Literal (annotatie, alleen gebruikt voor referentieobjecten, individuals zoals collectie-elementen)</i>
<code>skos:notation</code>			<i>Subject heeft als code/afkorting Literal (annotatie)</i>
<code>skos:definition</code>			<i>Subject heeft als definitie Literal (definitie zonder bron-referentie) (annotatie)</i>
<code>rdfs:isDefinedBy</code>			<i>Subject is gedefinieerd door Literal (definitie met bron-referentie) (annotatie)</i>
<code>rdfs:comment</code>			<i>Subject heeft als commentaar Literal (annotatie)</i>
<code>owl:inverseOf</code>			<i>Subject-property is de inverse van Object-property</i>
<code>hasUnit</code>		<code>owl:AnnotationProperty</code>	<i>Subject heeft als eenheid Literal (annotatie)</i>
<code>hasDateStart</code>		<code>owl:AnnotationProperty</code>	<i>Subject heeft als begindatum Literal (annotatie)</i>
<code>hasDateChange</code>		<code>owl:AnnotationProperty</code>	<i>Subject heeft als wijzigingsdatum Literal (annotatie)</i>
<code>hasAspect</code>	<code>isAspectOf</code>	<code>owl:ObjectProperty</code>	CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasAspect 0-n maal of min 0-n en max 1-n maal voorkomen
<code>hasValue</code>		<code>owl:DatatypeProperty</code> <code>owl:FunctionalProperty</code>	CE beschrijft restrictie op object: Bij subject met property hasValue mogen alleen objecten van een bepaald datatype voorkomen. Bij het datatype kunnen vervolgens restricties op inhoud worden meegegeven.
<code>hasReference</code>		<code>owl:ObjectProperty</code> <code>owl:FunctionalProperty</code>	CE beschrijft restrictie op object: Bij subject met property hasReference mogen alleen objecten van een bepaalde klasse (collectie) voorkomen
<code>hasInput</code>	<code>isInputOf</code>	<code>owl:ObjectProperty</code>	CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasInput 0-n maal of min 0-n en max 1-n maal voorkomen
<code>hasOutput</code>	<code>isOutputOf</code>	<code>owl:ObjectProperty</code>	CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasOutput 0-n maal of min 0-n en max 1-n maal voorkomen
<code>hasPart</code>	<code>isPartOf</code>	<code>owl:ObjectProperty</code>	CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasPart 0-n maal of min 0-n en max 1-n maal voorkomen
<code>hasConnection</code>		<code>owl:ObjectProperty</code> <code>owl:SymmetricProperty</code>	CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasConnection 0-n maal of min 0-n en max 1-n maal voorkomen
<code>hasRepresentation</code>		<code>owl:ObjectProperty</code> <code>owl:FunctionalProperty</code>	

Figuur 26 De toegepaste predicates in GWSW-OroX

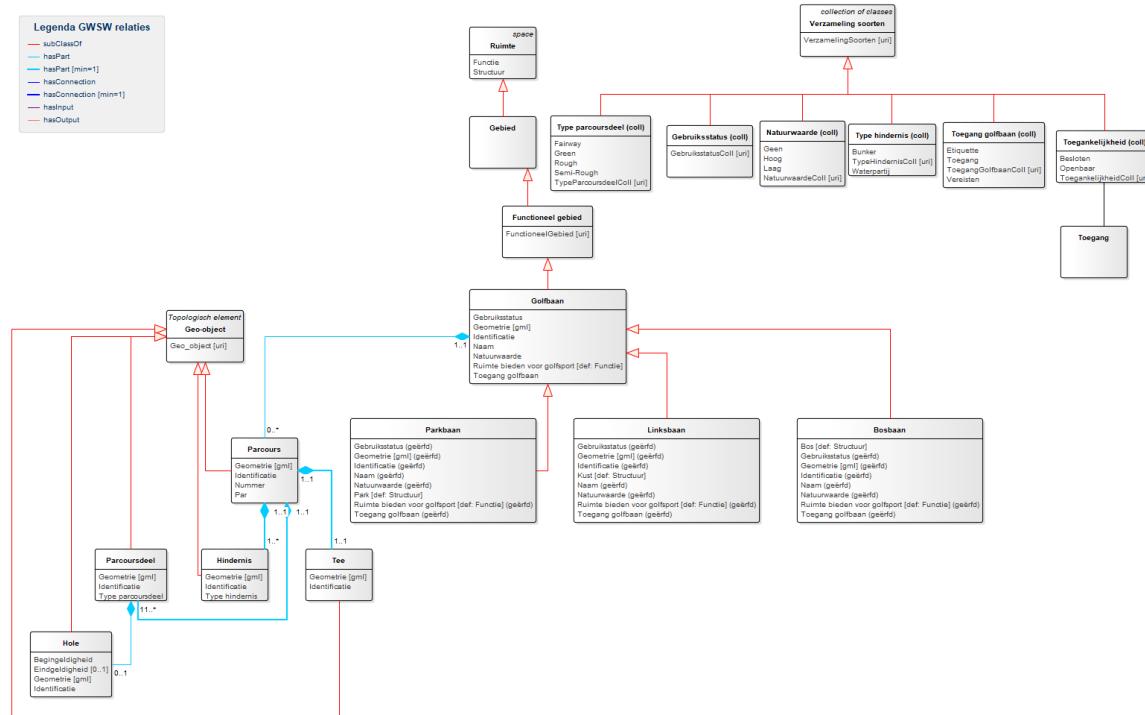
Voor het uitdrukken van class expressions voorziet OWL 2 in een groot aantal (restrictie) properties. Daarmee kunnen we klassen explicet onderscheiden. De GWSW Ontologie bevat de volgende:

Property	Toelichting
owl:onClass	Uitdrukken van cardinaliteit
owl:onProperty	Veelvuldig toegepast voor uitdrukken van klassen, vanwege "property central" principe.
owl:hasValue	Veelvuldig toegepast voor uitdrukken van klassen, vanwege "property central" principe.
owl:allValuesFrom	Uitdrukken van range bij waarden
owl:someValuesFrom	Uitdrukken van intrinsieke en onderscheidende kenmerken
owl:disjointWith	Uitdrukken van ruimtelijke hasPart relatie
owl:unionOf	Uitdrukken van ruimtelijke hasPart relatie
owl:qualifiedCardinality	Uitdrukken van verplichte properties
owl:maxQualifiedCardinality	Uitdrukken van maximum aantal properties
owl:minQualifiedCardinality	Uitdrukken van minimum aantal properties
owl:intersectionOf	Uitdrukken van onderscheidende kenmerken

Figuur 27 De toegepaste restricties in GWSW-OroX

Soortenbomen in de GWSW Ontologie zijn gebaseerd op de onderscheidende kenmerken van concepten. Door "determineren", maken van voldoende keuzes, wordt een soort gevonden. Onderscheidende kenmerken definiëren de concepten die vervolgens onderling gerelateerd worden in onder andere de soortenboom en samenstellingsboom.

Metamodel NEN 3610 conform GWSW-OroX



Figuur 28 Metamodel van NEN 3610 uitgedrukt als Linked Data in OroX profiel

7.4.4.3 UML-concepten versus OroX-concepten

Een eerste vergelijking tussen OroX en UML:

Relaties

UML-relatie	OroX-relatie
Generalisatie	rdfs:subClassOf, owl:Class
Aggregatie, compositie	gsws:hasPart
Algemene associaties	gsws:hasConnection, gsws:hasRepresentation

UML verdeelt de eigenschappen van klassen in een aantal rubrieken, in het OroX worden die "geklassificeerd". Zo worden UML-attributen aparte klassen die met relaties verbonden zijn aan de UML-kLASSE.

Klassen en attributen

UML-rubriek	OroX-relatie
Annotaties	rdfs:label, skos:altLabel, skos:notation, rdfs:isDefinedBy, rdfs:comment, ...
Attributen	gsws:hasAspect, gecombineerd met gsws:hasValue of gsws:hasReference
Methoden	gsws:hasInput, gsws:hasOutput

Met dit raamwerk is een transformatie van UML naar OroX relatief eenvoudig te realiseren. Dat kan nagenoeg (of geheel) automatisch, zie voor het resultaat van deze transformatie de gepubliceerde IMGOLF [ontologie](#) op [data.gsws.nl/1.4/imgolf](#).

7.4.4.4 Klassen

7.4.4.4.1 KLASSEN ALGEMEEN

De GWSW-OroX transformatie is identiek aan de eerder beschreven standaard. Een klasse is een instantie van [owl:Class](#). In GWSW-OroX heeft elke klasse de annotatie-properties [rdfs:label](#), [rdfs:comment](#), [skos:definition](#), [gsws:hasDateStart](#), [gsws:hasDateChange](#)

voorbeeld RDF representatie
<pre>imgolf:Parcours rdf:type owl:Class ; rdfs:label "Parcours"@nl ; rdfs:subClassOf imgolf:Geo_object ; skos:definition "Een van de speelvelden van een golfbaan"@nl ; rdfs:comment "Ook wel 'Hole' genoemd. Hier wordt de term Parcours gebruikt." ; gsws:hasDateStart "2017-11-16"^^xsd:date .</pre>

Een parcours als individu:

voorbeeld RDF representatie
<pre>:Hole2 a imgolf:Parcours; rdfs:label "Hole 2".</pre>

7.4.4.4.2 KLASSE MET STEREOTYPE «FEATURETYPE»

De GWSW-OroX transformatie is identiek aan de eerder beschreven standaard transformatie.

7.4.4.4.3 KLASSE MET STEREOTYPE «DATATYPE»

De GWSW-OroX transformatie past een «datatype» vaak toe. Ook om restricties te definiëren, bijvoorbeeld het parnummer ligt tussen 1 en 6:

voorbeeld RDF representatie

```
imgolf:Dt_wsw:Par
    rdfs:label          "Par (datatype)"@en ;
    rdf:type            rdfs:Datatype ;
    owl:equivalentClass
    [
        owl:onDatatype      xsd:integer ;
        rdfs:label           "minMaxValue_Kenmerk" ;
        owl:withRestrictions ([xsd:minInclusive "1"^^xsd:integer][xsd:maxInclusive "6"^^xsd:integer])
    ] .
```

7.4.4.4.4 KLASSE MET STEREOTYPE «UNION»

De GWSW-OroX transformatie past geen owl:unionOf toe. In GWSW-OroX wordt dit altijd een verzameling instanties:

voorbeeld RDF representatie

```
imgolf:NatuurwaardeColl
    rdf:type          owl:Class ;
    rdfs:label        "Natuurwaarde (coll)"@nl ;
    rdfs:subClassOf
    [
        rdf:type      owl:Class ;
        rdfs:label    "oneOf_Verzameling soorten" ;
        owl:oneOf     (imgolf:Hoog imgolf:Laag imgolf:Geen )
    ] .
```

7.4.4.4.5 KLASSE MET STEREOTYPE «EXTERNAL»

De GWSW-OroX transformatie is identiek aan de eerder beschreven standaard transformatie.

7.4.4.4.6 KLASSE MET STEREOTYPE «ENUMERATION»

De GWSW-OroX transformatie kent geen enumeraties (zie ook «union»), dit wordt altijd een verzameling instanties:

voorbeeld RDF representatie

```



```

7.4.4.4.7 KLASSE MET STEREOTYPE «CODELIST»

De GWSW-OroX transformatie kent geen «CodeList» (zie ook «union»), dit wordt een verzameling instanties. Als een code betekenisvol is, wordt deze ook gemodelleerd:

voorbeeld RDF representatie

```



```

7.4.4.5 Attributen

7.4.4.5.1 ATTRIBUTEN ALGEMEEN

In de GWSW-OroX transformatie wordt attributen ook klassen, Attributen worden toegewezen met de algemene relaties gsws:hasAspect en gsws:hasValue of gsws:hasReference.

voorbeeld RDF representatie

```



```

7.4.4.5.2 ATTRIBUUT MET STEREOTYPE «IDENTIFICATIE»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.5.3 ATTRIBUUT MET STEREOTYPE «VOIDABLE»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.5.4 ATTRIBUUT MET STEREOTYPE «MATERIELEHISTORIE»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.5.5 ATTRIBUUT MET STEREOTYPE «FORMELEHISTORIE»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.5.6 ATTRIBUUT MET STEREOTYPE «MATERIELELEVENSDUUR»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.5.7 ATTRIBUUT MET STEREOTYPE «FORMELELEVENSDUUR»

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.6 *Associaties*

Associaties, of relaties, zijn verbanden die gelegd worden tussen klassen in een UML model.

7.4.4.6.1 SIMPELE ASSOCIATIES

De GWSW-OroX transformatie hanteert `owl:ObjectProperty`, `owl:DatatypeProperty`, `owl:AnnotationProperty`, `owl:FunctionalProperty`

voorbeeld RDF representatie
<pre>gsws:hasAspect rdfs:label "has as aspect"@en ; rdf:type owl:ObjectProperty . gsws:hasValue rdfs:label "has as value"@en ; rdf:type owl:DatatypeProperty ; rdf:type owl:FunctionalProperty .</pre>

7.4.4.6.2 AGGREGATIE EN COMPOSITIE

De GWSW-OroX transformatie vereist expliciete deel-geheel relaties, zowel voor fysieke als ruimtelijk objecten.

voorbeeld RDF representatie

```

gsws:hasPart
    rdfs:label          "has as part"@en ;
    rdf:type            owl:ObjectProperty .

gsws:isPartOf
    rdfs:label          "has as part (inverse)"@en ;
    rdf:type            owl:ObjectProperty ;
    owl:inverseOf      gsws:hasPart .

:KoninklijkeHaagseGenCC rdf:type imgolf:Linksbaan ;
    rdfs:label          "Koninklijke Haagse Golf en Country Club"@nl ;
    gsws:hasPart        :DrivingRange;
    gsws:hasPart        :Hole1;
    gsws:hasPart        :Hole2.

```

7.4.4.6.3 GENERALISATIE/SPECIALISATIE

De GWSW-OroX transformatie vereist een expliciete soortenboom, klassen worden ingedeeld door hun onderscheidende kenmerken.

Bijvoorbeeld een imgolf:Tee is een subklasse van imgolf:Geo_object en onderscheidt zich doordat het de functie "afslagplaats" heeft:

voorbeeld RDF representatie

```

imgolf:Tee
    rdf:type          owl:Class ;
    rdfs:label        "Tee"@nl ;
    rdfs:subClassOf  imgolf:Geo_object ;
    skos:definition  "De afslagplaats aan het begin van een hole"@nl ;
    rdfs:comment     "Ook wel tee-box genoemd" ;
    rdfs:subClassOf [
        rdf:type          owl:Restriction ;
        owl:onProperty   gsws:hasAspect ;
        owl:someValuesFrom [
            [
                owl:intersectionOf ( gsws:Functie
                    [
                        rdf:type          owl:Restriction ;
                        owl:onProperty   gsws:hasReference ;
                        owl:hasValue      imgolf:Afslagplaats ] ) ]
        ] .
        imgolf:Afslagplaats
        rdf:type          gsws:Functie .
    ]

```

7.4.4.6.4 ASSOCIATIEKLASSEN

De GWSW-OroX transformatie volgt de eerder beschreven standaard transformatie.

7.4.4.7 Kardinaliteit

De GWSW-OroX transformatie hanteert vaak cardinaliteits-regels om klassen te beschrijven, de cardinaliteitsvooraarden helpen bij het identificeren van de soort.

Bijvoorbeeld als "iets" een imgolf:Hindernis als deel heeft, is dat "iets" wellicht een imgolf:Parcours:

voorbeeld RDF representatie

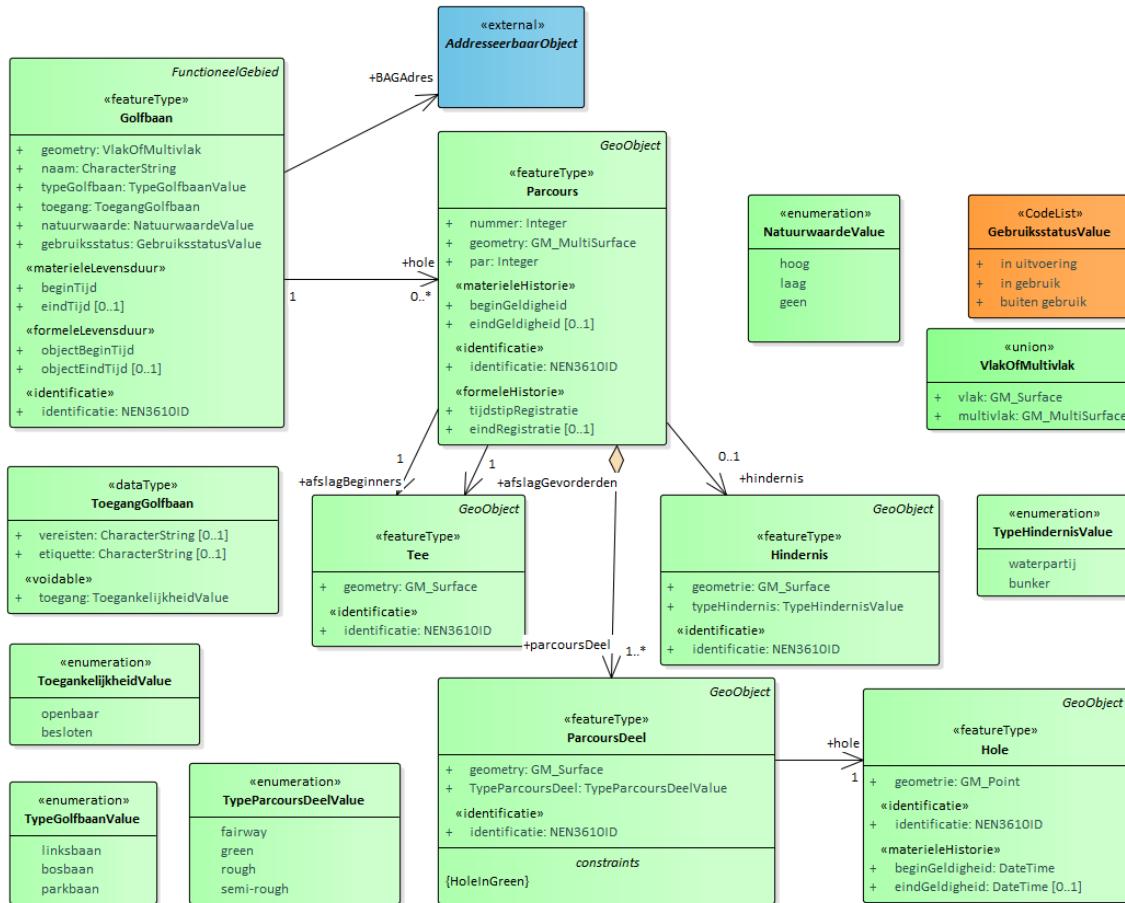
```
imgolf:Parcours
  rdfs:subClassOf
  [
    rdf:type          owl:Restriction ;
    rdfs:label        "wsw:hasPart_Hindernis" ;
    owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ;
    owl:onProperty   gsw:hasPart ;
    owl:onClass      imgolf:Hindernis
  ] .
```

8. Voorbeeld: UML-Golfbaan naar vocabulaire

Dit onderdeel is niet normatief.

Aan de hand van een informatiemodel in NEN 3610 conform UML van een golfbaan, IMGolf, kunnen we een Linked Data model opstellen.

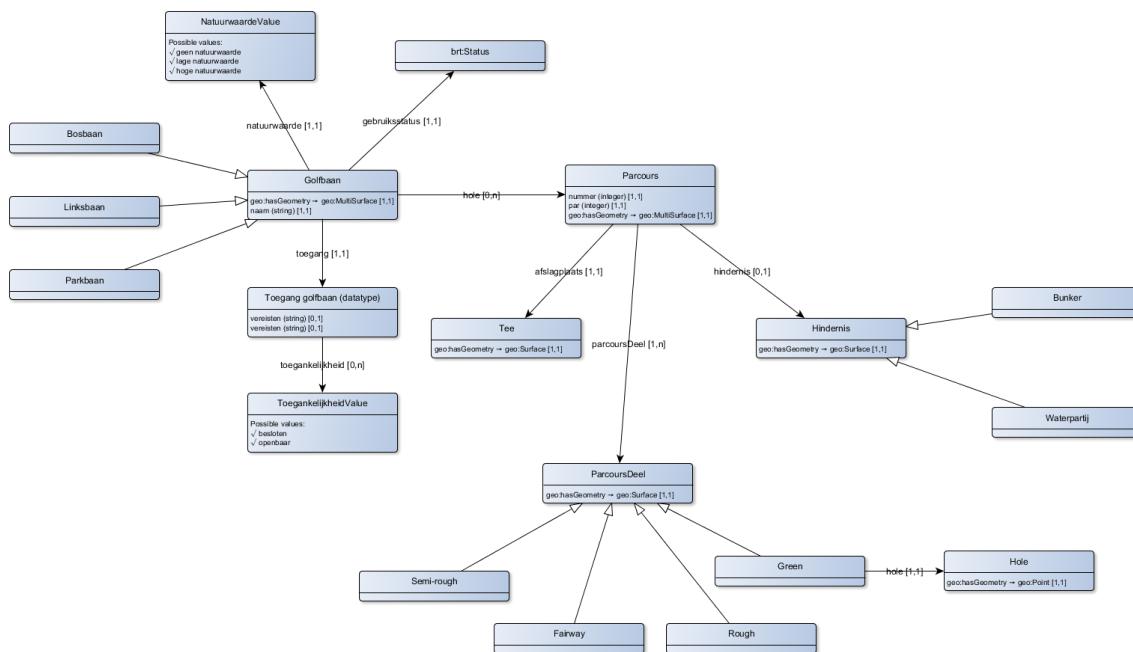
Elk van de drie stijlen leidt daarbij tot een eigen interpretatie van het IMGolf model. In dit hoofdstuk is voor elk van de drie stijlen het IMGolf UML getransformeerd naar een ontologie. In het tot standkomen van dit document was dit de eerste initiele transformatie die elke stijl toepaste op IMGolf. Deze transformatie leidt tot begrip over het verschil in de verschillende stijlen. Dit begrip is gebruikt om in hoofdstuk 7 de stijl specifieke transformatieregels op te stellen. De drie stijlen gebruiken elk een eigen notatiemodus voor modelconstructies. De diagrammen hebben wat dat betreft een eigen legenda. In § 9. [Aanpak voor het vergelijking van Linked Data datasets](#) worden de modellen wel uitgedrukt in één metamodel en is er één gezamenlijk diagramtype waarin de modellen gevisualiseerd worden. Het bronmodel in de UML naar LD transformatie is IMGolf. Onderstaand het UML klassendiagram hiervan.



Figuur 29 Model van Golfbaan (IMGolf) uitgedrukt in NEN 3610-UML

8.1 Model van data Golfbaan conform: DSO

Het Linked Data model conform DSO volgt de W3C standaarden RDF, RDFS, OWL en SHACL. Voor de visualisatie wordt aangesloten op de [Nederlandse best practice](#).

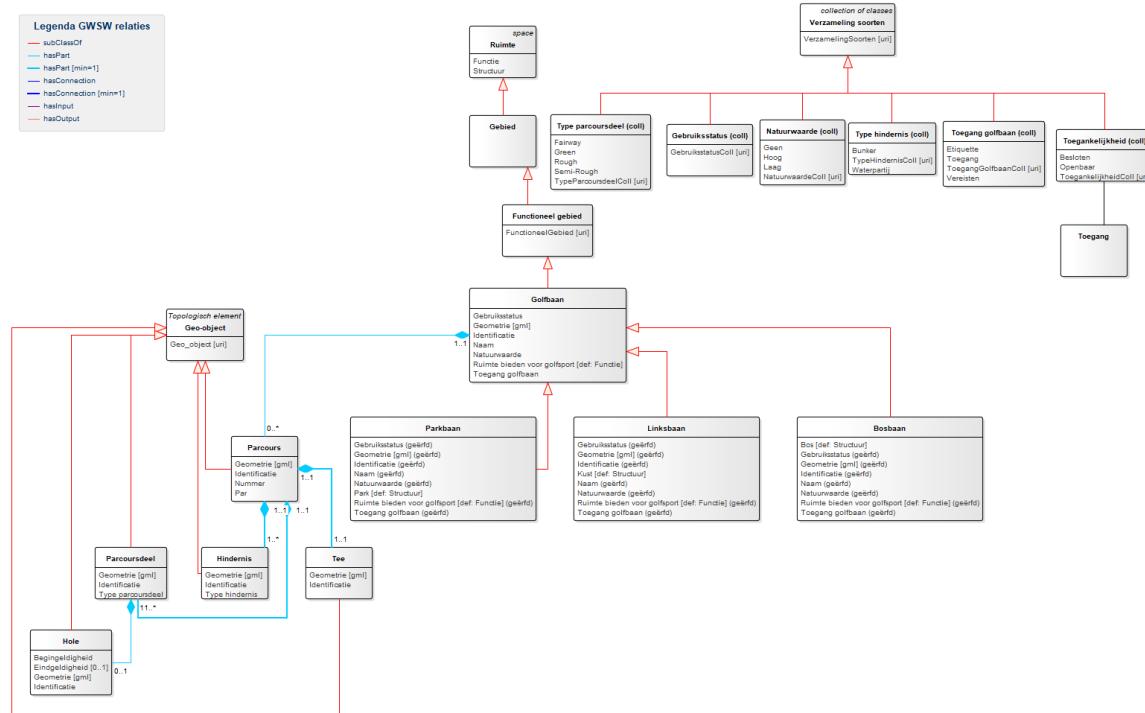


Figuur 30 Model van Golfbaan uitgedrukt als Linked Data in DSO profiel

8.2 Model van data Golfbaan conform: OroX

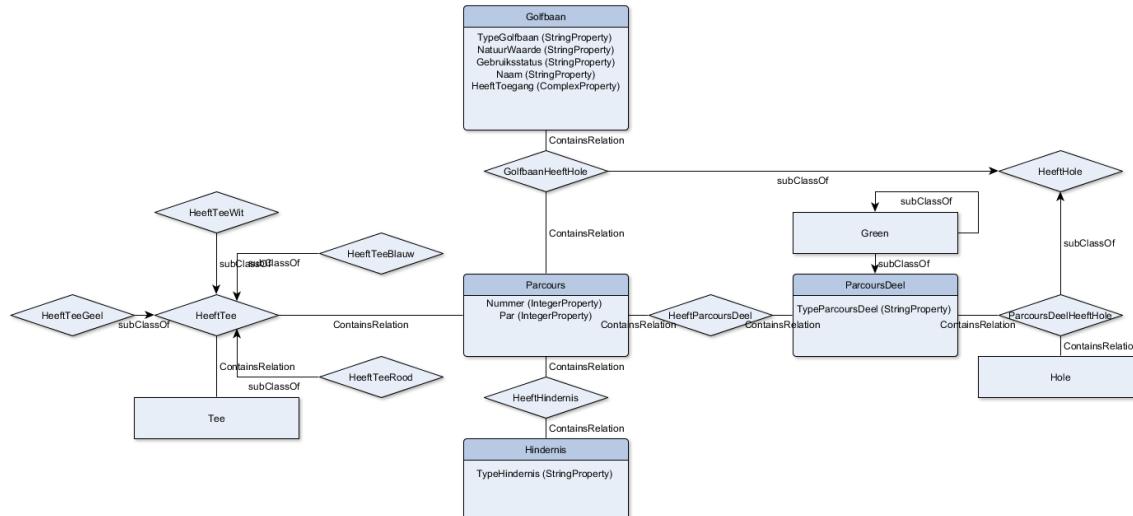
Er is applicatie om het OroX-protocol in een UML-schema te presenteren. Zonder handmatige ingreep in de transformatie ziet zo'n UML er natuurlijk anders uit, maar het beschrijft mogelijk wel het gezamenlijke UoD (zie hst. [Transformatie](#)).

Als proef is IMGOLF gemodelleerd in OroX conform de beschreven uitgangspunten voor het GWSW. Vervolgens is deze ontologie automatisch omgezet in een [OroX-presentatie in UML](#).



Figuur 31 Model van Golfbaan uitgedrukt als Linked Data in GWSW.orox profiel

8.3 Model van data Golfbaan conform: COINS



Figuur 32 Model van Golfbaan uitgedrukt als Linked Data in COINS profiel

9. Aanpak voor het vergelijking van Linked Data datasets

Dit onderdeel is niet normatief.

Een voordeel van Linked Data datasets is de mogelijkheid om dergelijke datasets met elkaar te verbinden en met elkaar te vergelijken. Door de eenduidige opslagstructuur met triples en het uniforme gebruik van RDF en RDFS, is het mogelijk om uit elke willekeurige Linked Data dataset een model af te leiden die een getrouwe weergave is van deze dataset. Dergelijke modellen kunnen vervolgens met elkaar worden vergeleken. Zo kan snel inzicht verkregen worden waaruit een willekeurige Linked Data dataset bestaat, en bovendien is het mogelijk om verschillende Linked Datasets met elkaar te vergelijken op modelniveau.

Dit hoofdstuk beschrijft de aanpak daartoe. We hebben deze aanpak gebruikt om inzicht te krijgen in hoeverre de drie stijlen van Linked Data modellering van elkaar verschillen en overeen komen.

9.1 Generatie opzet

Bij de generatie wordt het originele databestand opgepakt, en vervolgens wordt een model gegenereerd door de volgende stappen uit te voeren:

1. Indien de data een subclassificatie bevat, dan wordt deze overgenomen in het model en wordt de superklasse ook in het model gezet.
2. Klassen (**owl:Class**) worden aangemaakt voor elk unieke object op de positie **?subject rdf:type ?object**
3. Eigenschappen worden aangemaakt voor elk unieke predicate op de positie **?subject ?predicate ?object**
 - o Een eigenschap **owl:ObjectProperty** wordt aangemaakt indien het ?object een resource is (IRI en/of blank node)
 - o Een eigenschap **owl:DatatypeProperty** wordt aangemaakt indien het ?object een literal is
4. Een **rdfs:label** element wordt toegevoegd indien deze nog niet aanwezig was (afgeleid uit de URI van een klasse of eigenschap: het deel na de '#' of na de laatste '/')
5. Een nodeshape (**sh:NodeShape**) wordt aangemaakt voor elke gevonden klasse
6. Een datatype propertyshape (**sh:PropertyShape**) wordt aangemaakt voor elke gevonden datatype-eigenschap *per klasse*
 - o Het datatype wordt afgeleid uit het datatype van het ?object. Dit kunnen (dus) meerdere datatypes zijn
7. Een class propertyshape (**sh:PropertyShape**) wordt aangemaakt voor elke gevonden object-eigenschap *per klasse*
 - o Dit geldt alleen indien het ?object een IRI is (dus geen blank node)
 - o De class wordt afgeleid uit het type van het ?object (dwz: de triple ?object rdf:type ?type). Dit kunnen (dus) meerdere classes zijn, en het is ook mogelijk dat er geen class is
 - o Een class propertyshape kun je zien als een relatie tussen twee klassen
8. Een blank node propertyshape (**sh:PropertyShape**) wordt aangemaakt voor elke gevonden object-eigenschap *per klasse*
 - o Dit geldt alleen indien het ?object een Blank node is
 - o De class wordt afgeleid uit het type van het ?object (dwz: de triple ?object rdf:type ?type). Dit kunnen (dus) meerdere classes zijn, en het is ook mogelijk dat er geen class is
 - o Een blank node propertyshape kun je zien als een complexe eigenschap bij een klasse
9. Een minimale cardinaliteit (**sh:minCount**) wordt vastgesteld door te tellen hoevaak een bepaalde eigenschap voorkomt. Is er minimaal één exemplaar van een specifieke klasse die een eigenschap *niet* heeft, dan is de minimum cardinaliteit 0.

10. Een maximale cardinaliteit (**sh:maxCount**) wordt vastgesteld door te tellen hoevaak een bepaalde eigenschap voorkomt bij een specifieke klasse. Het totaal aantal wordt de maximum cardinaliteit (dit getal is feitelijk alleen zinvol voor inzicht in de huidige dataset. Meestal zal een maximale waarde van meer dan 1 betekenen dat er feitelijk geen maximum is).
11. Indien een exemplaar van een klasse ook altijd een exemplaar is van een andere klasse, dan wordt dit weergegeven middels de "implementeert" relatie (door een propertyshape waarbij veronderstelt wordt dat **rdf:type** altijd beide klassen omvat).

Vanzelfsprekend kan alleen maar die dingen gegenereerd worden die ook in de brondaten zitten. Uitleg of afwijkende namen zijn niet mogelijk, en ook kunnen cardinaliteiten of subklassificaties niet met zekerheid worden afgeleid.

Een gegenereerde IMGolf model zal afwijken van een met de hand opgestelde IMGolf model. Het verschil geeft mooi aan wat je als modelleur nog toevoegd als uitleg.

Ook is het niet mogelijk om deze aanpak te gebruiken voor de UML / GML data, aangezien dit geen Linked Data is en dus niet generiek in te lezen is (hier zou dus eerst een basale vertaling van GML naar Linked Data gemaakt moeten worden).

9.2 Modelvergelijking

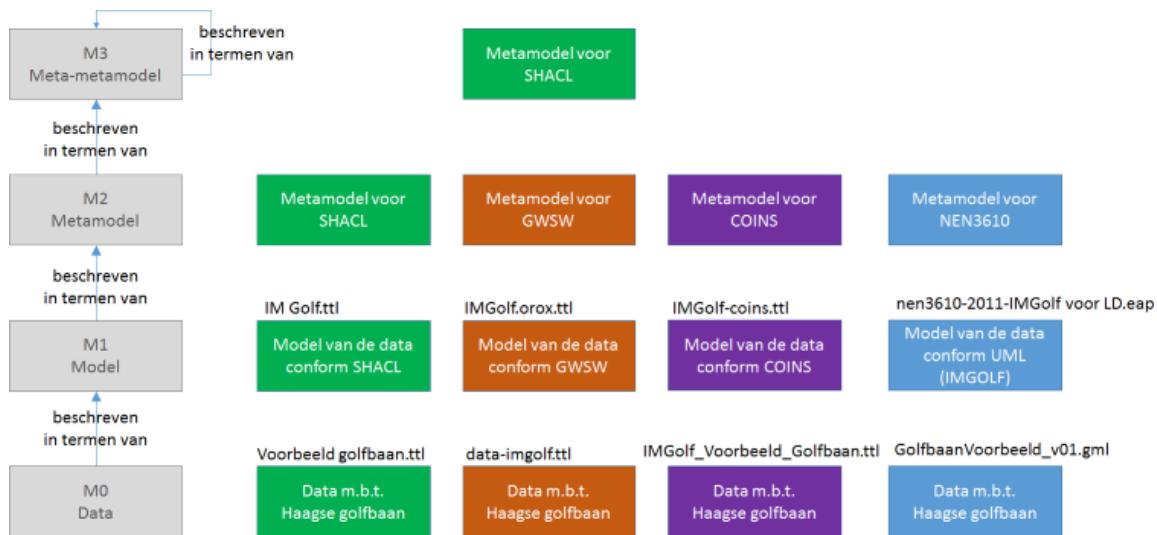
Linked Data op het niveau van de concrete data (zoals de voorbeelddata over de Koninklijke Haagse Golf & Countryclub) noemen we Linked Data op niveau M0. Het model dat we hieruit kunnen afleiden op basis van het mechanisme in de voorgaande sectie noemen we M1.

Dergelijke Linked Data modellen zijn zelf ook weer Linked Data. Het is dan ook mogelijk om het mechanisme te herhalen:

- Vanuit M0 het model te genereren dat een correcte beschrijving geeft van de structuur van de data
- Vanuit M1 het metamodel te genereren dat een correcte beschrijving geeft van de structuur van het model
- Vanuit M2 het metametamodel te genereren dat een correcte beschrijving geeft van de structuur van het metamodel

Voor de vergelijking maken we gebruik van een voorbeeld: de concrete informatie met betrekking tot de golfbaan van de Koninklijke Haagse Golf & Countryclub. De vier modelniveau's zijn dan als volgt:

- M0 betreft het niveau met concrete data: dus in ons voorbeeld data over de Koninklijke Haagse Golf & Countryclub
- M1 betreft het niveau met het model van de data. In ons voorbeeld zijn dit de vier modellen: het origineel in UML en de drie Linked Data stijlen van COINS, OroX en DSO.
- M2 betreft het niveau met de metamodelen van de data. Hiermee kun je dus zien wat de verschillen zijn tussen de vier modellen, en is het mogelijk om vertalingen te maken van het ene model naar het andere model
- M3 betreft het niveau met het metametamodel. Als metametamodel hebben we gekozen voor het model dat afkomstig is van het mechanisme uit de vorige sectie, SHACL.



Van de golfbaan hebben we zowel vier verschijningsvormen op het niveau van M0 (de concrete data) als van M1 (het model):

- *GolfbaanVoorbeeld_v01.gml* betreft de GML representatie (M0) conform NEN 3610 zoals gemodelleerd in *nen3610-2011-IMGolf voor LD.eap* (M1)
- *Voorbeeld golfbaan.ttl* betreft een DSO representatie (M0) conform SHAACL/OWL/RDFS zoals gemodelleerd in *IM Golf.ttl* (M1)
- *data-imgolf.ttl* betreft een COINS uitwisselingsbestand (M0) zoals gemodelleerd in *IMGolf-coins.ttl* (M1)
- *IMGolf_Voorbeeld_Golfbaan.ttl* betreft een GWSW uitwisselingsbestand (M0) zoals gemodelleerd in *IMGolf.orox.ttl* (M1)

9.3 Linked Data modelstijlen

9.3.1 Huidige stijlen in gebruik

Binnen Nederland zijn er op dit moment drie stijlen actief als het gaat om de representatie van NEN 3610 informatie als Linked Data:

- GSW Ontologie in RDF: OroX-protocol voor het Gegevenswoordenboek Stedelijk Water
- COINS: uitwisselingsstandaard voor BIM informatie
- DSO: beschrijving t.b.v. het DSO op basis van de W3C specificaties SHACL/OWL/RDFS

Concreet betekent dit, dat er feitelijk drie mogelijke targets zijn (RDF bestanden) voor één source (een UML model of XML/GML bestand conform NEN 3610). In totaal betreft dit vier verschijningsvormen van dezelfde informatie. Wel geldt dat de drie Linked Data representaties makkelijker met elkaar zijn te vergelijken en aan elkaar zijn te relateren, aangezien in al deze gevallen sprake is van Linked Data. Dit betekent dat vanuit het ene model verwezen kan worden naar het andere model, bijvoorbeeld met een eigenschap rdfs:seeAlso, of met een eigen gedefinieerde eigenschap:

ex:vergelijkbaarMetModelUitAnderModel. Niet altijd is namelijk precies sprake van exact dezelfde elementen. In geval van geometrische objecten zoals bij NEN 3610 zou ook gebruik kunnen worden gemaakt van de geometrische relaties uit de Geosparql vocabulaire. Zie [Spatial Knowledge Graph](<https://www.mdpi.com/2078-2489/10/10/310>) voor een toepassing hiervan.

Voor een deel zijn deze verschillen verklaarbaar doordat er bij elke stijl een verschillende focus is op beoogde doelen of use cases die bediend moeten worden. Deze verschillen uiten zich onder andere in het gebruik van verschillende Linked Data

vocabulaires RDF, RDFS, SKOS, OWL en SHACL. Voor elke stijl wordt in onderstaande matrixen de use cases en de gerelateerde vocabulaires aangegeven.

Stijl: COINS	vocabulaires				
use cases	RDF	RDFS	OWL	SKOS	SHACL
uitwisselen informatiemodel (bim info)	✓	✓	✓		
heterogene informatie in samenhang	✓	✓	✓		
kennismodel (ontologie, semantiek)	✓	✓	✓		
data validatie			✓		

COINS is ontwikkeld in een periode dat de SHACL vocabulaire nog niet beschikbaar was. Dit verklaart het gebruik van OWL voor validatie.

Stijl: OroX	vocabulaires				
use cases	RDF	RDFS	OWL	SKOS	SHACL
begrippenkader (vocabulaire, thesaurus)	✓	✓		✓	
kennismodel (ontologie, semantiek)	✓	✓	✓		
web van data (publicatie)	✓	✓	✓		
data validatie			✓		

Het OroX gebruikt de class expressions in OWL-2 om klassen zo expliciet mogelijk te beschrijving. Validatie van datasets vindt plaats met SPARQL-queries, tijdens de ontwikkeling van het OroX was de SHACL vocabulaire nog niet beschikbaar.

Stijl: DSO	vocabulaires				
use cases	RDF	RDFS	OWL	SKOS	SHACL
uitwisselen informatiemodel	✓	✓	✓	✓	✓
begrippenkader (vocabulaire, thesaurus)				✓	
kennismodel (ontologie, semantiek)	✓	✓	✓		
web van data (publicatie)	✓	✓	✓		
data validatie					✓
kennisafleiden, artificial intelligence			✓		

9.3.2 NEN 3610 use cases en requirements

Op basis van de NEN 3610 use cases, is ook vast te stellen welke vocabulaires het beste gebruikt kunnen worden voor een basis NEN 3610-stijl.

Stijl: NEN 3610	vocabulaires				
use cases	RDF	RDFS	OWL	SKOS	SHACL
uitwisselen informatiemodel	✓	✓	✓	✓	✓
waardelijsten				✓	
begrippenkader (vocabulaire, thesaurus)				✓	
kennismodel (ontologie, semantiek)	✓	✓	✓		
data publicatie	✓	✓			

data validatie					✓
vocabulaires in samenhang/harmoniseren				✓	✓

9.4 Modelvergelijking op dataniveau

9.4.1 Inleiding

De Koninklijke Haagse Country & Golfclub golfbaan is als voorbeeld data opgepakt. Deze voorbeeld data is in vier formaten beschikbaar:

- [GML](#), Een XML voorbeeldbestand conform het XSD dat hoort bij het [UML](#) IMGolf informatiemodel
- COINS: Een Linked Data voorbeeldbestand conform de COINS uitwisselingsstandaard
- GWSW-OROX: Een Linked Data voorbeeldbestand conform de GWSW-OROX uitwisselingsstandaard
- W3C RDF/OWL: Een Linked Data voorbeeldbestand conform de [W3C](#) RDF en OWL vocabulaires, zonder aanvullende afspraken

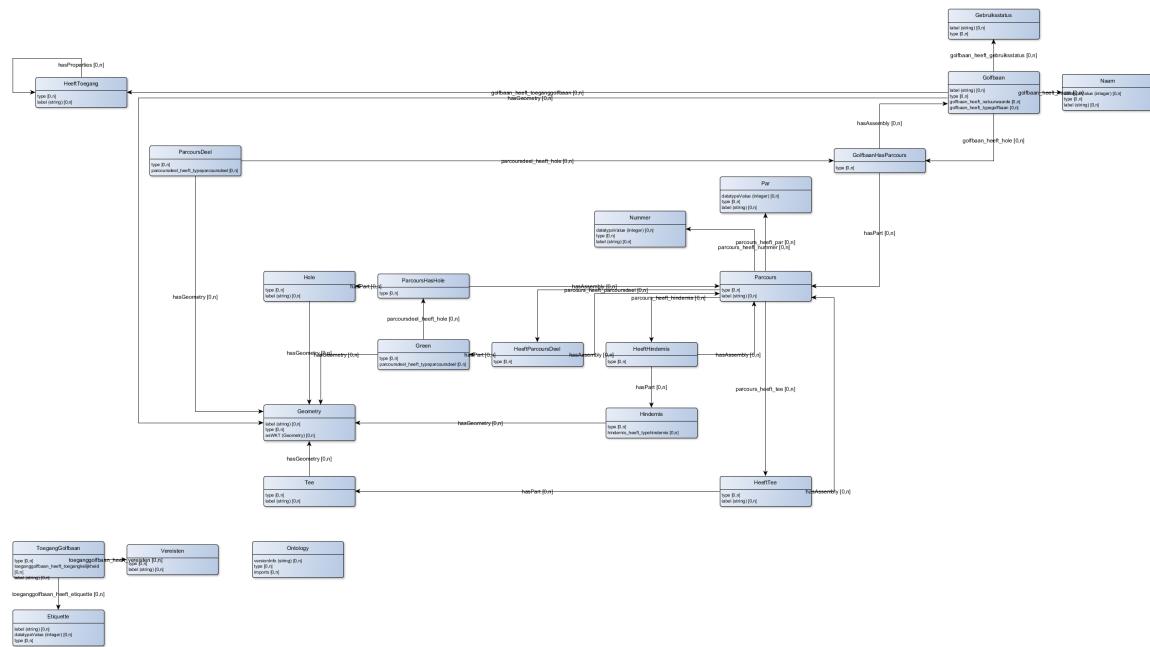
Door deze opzet, kunnen we een goed beeld krijgen wat de verschillen op data-niveau zijn als je deze verschillende stijlen toepast.

Het vergelijken van deze stijlen is pas goed mogelijk als je de verschillende modellen van de data naast elkaar kunt houden. Dat is lastig, omdat deze vanzelfsprekend van elkaar verschillen: ze kennen een eigen [metamodel](#) (respectievelijk dat van UML, COINS, GWSW-OROX en W3C)

Wel is het mogelijk vanuit de data, in het geval van de Linked Data situatie, voor elk van de stijlen een model te genereren. Hierdoor zijn de verschillen goed zichtbaar

9.4.2 Gegenereerd COINS model

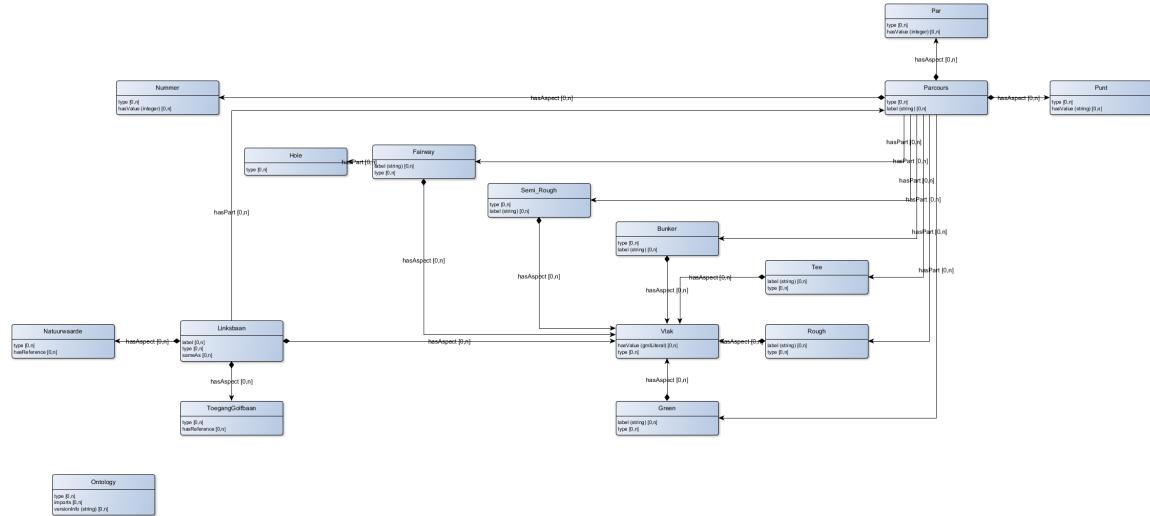
Onderstaand diagram is de visualisatie van het RDF/OWL/SHACL model dat gegenereerd is conform bovenstaande aanpak op basis van het COINS voorbeeldbestand van de Koninklijke Haagse Golf & Country club



Figuur 33 afgeleid Model van Golfbaan vanuit het COINS uitwisselingsbestand

9.4.3 Gegenereerd GWSW-OROX model

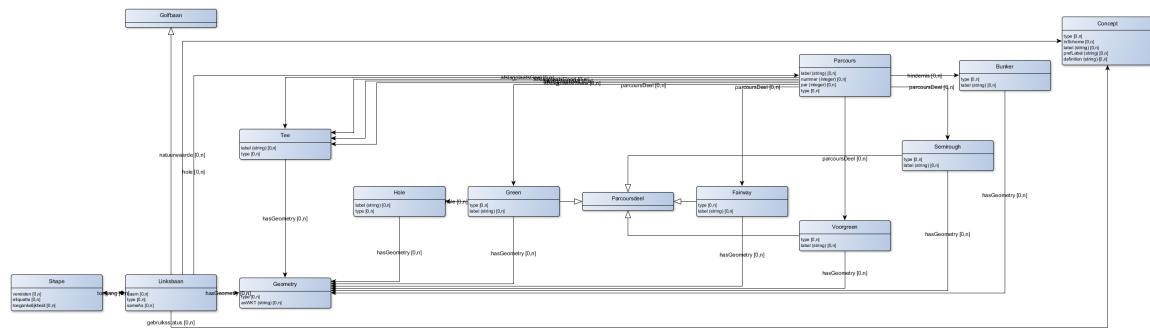
Onderstaand diagram is de visualisatie van het RDF/OWL/SHACL model dat gegenereerd is conform bovenstaande aanpak op basis van het GWSW-OROX voorbeeldbestand van de Koninklijke Haagse Golf & Country club



Figuur 34 afgeleid Model van Golfbaan vanuit het OroX uitwisselingsbestand

9.4.4 Gegenereerd W3C model

Onderstaand diagram is de visualisatie van het RDF/OWL/SHACL model dat gegenereerd is conform bovenstaande aanpak op basis van het W3c RDF/OWL voorbeeldbestand van de Koninklijke Haagse Golf & Country club



Figuur 35 afgeleid Model van Golfbaan vanuit de DSO dataset

A. Glossary

Dit onderdeel is niet normatief.

Beschouwingsgebied: beeld van de echte of hypothetische wereld dat binnen de context van een domein alles van belang omvat [[NEN3610](#)].

Unique Name Assumption (UNA): de aanname dat een ding maar één naam (id) heeft en dat verschillende namen dus verwijzen naar verschillende dingen in de werkelijkheid [[UNA](#)].

Reificatie: een manier om metadata aan een [triple](#) te hangen [[REIFICATION](#)].

UML: Unified Modeling Language [[uml](#)]

GML: Geography Markup Language [[gml](#)], een op XML gebaseerde standaard voor uitwisseling van geografische data tussen systemen.

RDF: Resource Description Framework. Zie [[rdf11-primer](#)].

RDFS: RDF Schema, taal waarin je een Linked Data ontologie kan uitdrukken [[rdf-schema](#)].

SKOS: Simple Knowledge Organization System. Zie [[skos-primer](#)]

OWL: Web Ontology Language [[owl2-overview](#)], bevat uitgebreidere mogelijkheden dan RDFS om een Linked Data ontologie uit te drukken.

SHACL: Shapes Constraints Language [[shacl](#)], een taal om RDF data te valideren tegen een set regels.

PROV-O: The PROV Ontology [[prov-o](#)], een taal om versie- en herkomstinformatie te beschrijven.

Ontologie: een kennismodel van een specifiek kennisdomein in de werkelijkheid [[ONTOLOGY](#)]. Bevat een set regels, die gebruikt kunnen worden om extra kennis af te leiden uit gelinkte data. Met behulp van zo'n model kunnen computers begrijpen wat de data betekent en redeneren over data.

Vocabulaire: Een verzameling van termen waarmee je je uitdrukt. De term vocabulaire wordt ook wel gebruikt in bredere zin of als synoniem van de term ontologie (maar dit laatste niet binnen dit document).

Begrippenkader: Een model van begrippen, dat beschrijft welke concepten er in het domein dat je wilt modelleren een rol spelen, en wat ze betekenen [[vocab](#)]. De begrippen zijn (tekstueel) gedefinieerd en hebben onderlinge relaties, zoals bijvoorbeeld in een thesaurus.

Feature type: Objecttype gebruikt voor het representeren van geo-informatie [[NEN3610](#)].

Turtle (ttl): RDF serialisatieformaat. Zie [[turtle](#)]

RDF/XML: RDF serialisatieformaat gebaseerd op XML.

WFS: Web Feature Service (WFS) is een gestandaardiseerd protocol waarmee je geografische vector informatie kunt opvragen, aanleveren, bewerken en analyseren [[wfs](#)].

Object Oriëntatie: Een paradigma waarbij een systeem wordt opgebouwd uit objecten. Uitgangspunt bij het objectgeoriënteerd modelleren is dat de werkelijkheid is opgebouwd uit objecten (real world objects). Eigenschappen van objecten worden beschreven door kenmerken. Het model is een abstracte representatie van de werkelijkheid. Objecten worden in het model geo-objecten genoemd (geographic features) en de eigenschappen worden beschreven door attributen. Een objectgeoriënteerd informatiemodel is objectgericht: het geeft informatie over individueel te onderscheiden objecten binnen de beschreven werkelijkheid. Het object is de eenheid van informatie [[NEN3610](#)].

Representational State Transfer (REST): Een manier om APIs op te bouwen. Het belangrijkste principe van REST is het scheiden van de API in logische resources ("dingen"). De resources beschrijven de informatie van het "ding". Deze resources worden gemanipuleerd met behulp van HTTP-verzoeken en HTTP-operaties. Elke operatie (GET, POST, PUT, PATCH, DELETE) heeft een specifieke betekenis [[rest](#)].

Application Programming Interface (API): een combinatie van technische bestanden, documentatie en andere ondersteuning die helpen bij het aanroepen van externe applicaties [[api](#)].

Triple: een bewering die invulling geeft aan één specifieke eigenschap van een onderwerp [[rdf11-primer](#)]. Naast een subject bestaat elke bewering of triple uit een eigenschap en een waarde. De waarde kan ook weer een onderwerp op zich zijn met een eigen unieke sleutel en eigenschappen.

Internationalized Resource Identifier (IRI): een speciale vorm van een URI, die beter internationaal bruikbaar is omdat je in een IRI ook tekens uit niet-latijns schrift kan gebruiken [[IRI](#)].

Blank node: Een knoop in een RDF graaf die geen waarde of IRI identifier heeft [[rdf11-primer](#)].

Subject: in Linked Data is dit het eerste component van een triple, namelijk het onderwerp van de bewering [[rdf11-primer](#)].

Non-information resource: een ding in de werkelijkheid; het tegenovergestelde van een information resource.

Information resource: een document of record over een ding uit de werkelijkheid. [[webarch](#)] definieert de term Information resource als "A resource which has the property that all of its essential characteristics can be conveyed in a message."

Normalisatie: een modelleervorm waarbij het model zo dicht mogelijk bij de te beschrijven werkelijkheid ligt.

Setoriëntatie: Een leer waarbij dingen worden ingedeeld in sets met gedeelde eigenschappen. Deze sets kunnen hiërarchisch gerelateerd worden en kunnen overlappen. Synoniem: verzamelingenleer [[verzamelingenleer](#)].

Multiple inheritance: meervoudige overerving, waarbij een klasse meerdere superklassen kan hebben. In andere woorden: een set kan onderdeel zijn van meerdere andere sets.

Metamodel: een set regels voor het modelleren van informatie. Voorbeelden: het General Feature Model [[iso-19109-2015](#)], het Metamodel voor Informatiemodellen [[MIM](#)].

Supertype: de hogere klasse in een overervingshiërarchie [[NEN3610](#)].

Union: gestructureerd datatype zonder identiteit waarvan precies één van de eigenschappen aanwezig is in elke instantie [[NEN3610](#)].

Datatype: UML constructie met het stereotype «datatype»: gestructureerd datatype zonder identiteit [[NEN3610](#)]. Specificatie van een waardedomein met operaties die zijn toegestaan op waarden uit dit domein [[iso-19103-2015](#)]

Open world assumption (OWA): modelleeruitgangspunt waarbij het model afleidt wat mogelijk is (bijvoorbeeld in OWL).

Closed world assumption (CWA): modelleeruitgangspunt waarbij het model beperkt wat mogelijk is (bijvoorbeeld in UML).

W3C: World Wide Web Consortium, internationale organisatie voor de standaardisatie van het web.

OGC: Open Geospatial Consortium, internationale organisatie voor de standaardisatie van geo-informatie.

OMG: Object Management Group, internationale organisatie voor de standaardisatie van technische standaarden zoals UML.

OTL: Object Type Library. Term die in de bouwsector gebruikt wordt en overeenkomt met Ontologie.

B. Referenties

B.1 Normatieve referenties

[geodcat-ap]

GeoDCAT-AP: A geospatial extension for the DCAT application profile for data portals in Europe. Version 1.0.1.

European Commission. 2 August 2016. URL: <https://joinup.ec.europa.eu/solution/geodcat-application-profile-data-portals-europe>

[NEN3610]

NEN 3610:2011 nl - Basismodel geo-informatie - Termen, definities, relaties en algemene regels voor de uitwisseling van informatie over aan de aarde gerelateerde ruimtelijke objecten. NEN. 2011-03-01. Definitief. URL: <https://www.nen.nl/NEN-Shop/Norm/NEN-36102011-nl.htm>

[semic-core]

e-Government Core Vocabularies. 2019. URL: <https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/core-vocabularies#Who%20is%20using%20Core%20Vocabularies>

B.2 Informatieve referenties

[api]

Application programming interface. Wikipedia. 2019-02-22. URL: https://en.wikipedia.org/wiki/Application_programming_interface

[CEN-TC-442]

CEN/TC 442 - Building Information Modelling (BIM). 2019. URL: https://standards.cen.eu/dyn/www/f?p=204:7:0:::FSP_ORG_ID:1991542&cs=16AAC0F2C377A541DCA571910561FC17F

[gml]

Geography Markup Language (GML) Encoding Standard. Open Geospatial Consortium Inc. URL: <http://www.opengeospatial.org/standards/gml>

[IMBAG]

Informatie Model Basisregistratie Adressen en Gebouwen. Geonovum. 2016-11-04. Definitief. URL: https://github.com/Geonovum/IMBAG/blob/master/concept%20catalogus/semantisch%20gegevensmodel/20161104_IMBAG_UML_concept.EAP

[INSPIRERDF]

Guidelines for the RDF encoding of spatial data. ARE3NA project "INSPIRE Re3ference Platform Phase 2". 2017-07-17. Draft. URL: <http://inspire-eu-rdf.github.io/inspire-rdf-guidelines>

[IRI]

Internationalized resource identifier. Wikipedia. 20180707. URL: https://nl.wikipedia.org/wiki/Internationalized_resource_identifier

[iso-19103-2015]

Geographic information -- Conceptual schema language. ISO/TC 211. ISO. 2015. International Standard. URL: <https://www.iso.org/standard/56734.html>

[iso-19109-2015]

Geographic information -- Rules for application schema. ISO/TC 211. ISO. 2015. International Standard. URL: <https://www.iso.org/standard/59193.html>

[iso-19126-2009]

ISO 19126:2009 Geographic information -- Feature concept dictionaries and registers. 200911. URL: <https://www.iso.org/standard/44875.html>

[iso-21597-1]

ISO/DIS 21597-1: Information container for data drop -- Exchange specification -- Part 1: Container. 2019. URL: <https://www.iso.org/standard/74389.html>

[LINKED-DATA-EVOLVING-WEB]

Linked Data: Evolving the Web into a Global Data Space (1st edition). Tom Heath; Christian Bizer. Morgan & Claypool. 2011. URL: <http://linkeddatabook.com/editions/1.0/>

[MIM]

MIM - Metamodel Informatie Modellering. Geonovum. 2017-06-14. Definitief. URL: <https://docs.geostandaarden.nl/mim/mim10/>

[MIM11]

MIM - Metamodel Informatie Modellering. Geonovum. 2019-10-29. Werkversie. URL: <https://docs.geostandaarden.nl/mim/wv-st-mim11-20191029/>

[ONTOLOGY]

Ontology (information science). Wikipedia. 2019-02-22. URL: [https://en.wikipedia.org/wiki/Ontology_\(information_science\)](https://en.wikipedia.org/wiki/Ontology_(information_science))

[OPM]

Ontology for Property Management. 2018-11-22. URL: <https://w3c-lbd-cg.github.io/opm/>

[OSLO-EA-RDF]

OSLO - Enterprise Architect RDF Conversion Tool. Informatie Vlaanderen. 2019-03-25. URL: <https://github.com/Informatievlaanderen/OSLO-EA-to-RDF>

[owl2-overview]

OWL 2 Web Ontology Language Document Overview (Second Edition). W3C OWL Working Group. W3C. 11 December 2012. W3C Recommendation. URL: <https://www.w3.org/TR/owl2-overview/>

[owl2-primer]

OWL 2 Web Ontology Language Primer (Second Edition). Pascal Hitzler; Markus Krötzsch; Bijan Parsia; Peter Patel-Schneider; Sebastian Rudolph. W3C. 11 December 2012. W3C Recommendation. URL: <https://www.w3.org/TR/owl2-primer/>

[prov-o]

PROV-O: The PROV Ontology. Timothy Lebo; Satya Sahoo; Deborah McGuinness. W3C. 30 April 2013. W3C Recommendation. URL: <https://www.w3.org/TR/prov-o/>

[QUDT]

QUDT - Quantities, Units, Dimensions and Data Types Ontologies. Ralph Hodgson; Paul J. Keller; Jack Hodges; Jack Spivak. 18 March 2014. URL: <http://www.qudt.org/>

[rdf-schema]

RDF Schema 1.1. Dan Brickley; Ramanathan Guha. W3C. 25 February 2014. W3C Recommendation. URL: <https://www.w3.org/TR/rdf-schema/>

[rdf11-primer]

RDF 1.1 Primer. Guus Schreiber; Yves Raimond. W3C. 24 June 2014. W3C Note. URL: <https://www.w3.org/TR/rdf11-primer/>

[REIFICATION]

Reification (computer science). Wikipedia. 2019-02-22. URL: [https://en.wikipedia.org/wiki/Reification_\(computer_science\)#RDF_and_OWL](https://en.wikipedia.org/wiki/Reification_(computer_science)#RDF_and_OWL)

[rest]

Representational state transfer. Wikipedia. 2019-02-22. URL: https://en.wikipedia.org/wiki/Representational_state_transfer

[sdw-bp]

Spatial Data on the Web Best Practices. Jeremy Tandy; Linda van den Brink; Payam Barnaghi. W3C. 28 September 2017. W3C Note. URL: <https://www.w3.org/TR/sdw-bp/>

[shacl]

Shapes Constraint Language (SHACL). Holger Knublauch; Dimitris Kontokostas. W3C. 20 July 2017. W3C Recommendation. URL: <https://www.w3.org/TR/shacl/>

[skos-primer]

SKOS Simple Knowledge Organization System Primer. Antoine Isaac; Ed Summers. W3C. 18 August 2009. W3C Note. URL: <https://www.w3.org/TR/skos-primer/>

[turtle]

RDF 1.1 Turtle. Eric Prud'hommeaux; Gavin Carothers. W3C. 25 February 2014. W3C Recommendation. URL: <https://www.w3.org/TR/turtle/>

[uml]

OMG Unified Modeling Language. Open Management Group. OMG. 1 March 2015. Normative. URL: <http://www.omg.org/spec/UML/>

[UNA]

Unique name assumption. Wikipedia. 20181212. URL: https://en.wikipedia.org/wiki/Unique_name_assumption

[verzamelingenleer]

Verzamelingenleer. Wikipedia. 20181103. URL: <https://nl.wikipedia.org/wiki/Verzamelingenleer>

[vocab]

Controlled vocabulary. Wikipedia. 2019-02-22. URL: https://en.wikipedia.org/wiki/Controlled_vocabulary

[webarch]

Architecture of the World Wide Web, Volume One. Ian Jacobs; Norman Walsh. W3C. 15 December 2004. W3C Recommendation. URL: <https://www.w3.org/TR/webarch/>

[wfs]

Web Feature Service 2.0 Interface Standard. Panagiotis (Peter) A. Vretanos. OGC. 10 July 2014. OGC Interface Standard. URL: <http://www.opengeospatial.org/standards/wfs>

↑