# GWSW Ontologie in RDF

## *Een beschrijving van het protocol GWSW-OroX*

Van: Stichting RIONED

Versie historie
20200812: Modelleringsprincipes toegevoegd (hst 1.5)
20191030: Tabel met Top Level concepten toegevoegd (hst 2.3)
20190813: Toelichting op inrichting GWSW-OroX (hst 1.3)
20190521: Diagram met structuur GWSW Relaties toegevoegd
20190518: URL binnen rdfs:comment in apart predicate rdfs:seeAlso ondergebracht
20181003: Subtitel toegevoegd: dit is het GWSW-OroX protocol
20180305: Onderscheidende kenmerken uitgebreid voor activiteiten
20171028: Samenvatting completer gemaakt
20171023: Eerste gepubliceerde versie
20171013: Eerste opzet document

## Inhoud

# 1 Inleiding

Het W3C definieert standaarden voor het Semantisch Web met als basis de triple-vorm: de Subject-Predicate-Object constructie. Het basisprotocol dat hieraan ten grondslag ligt is RDF.

> De term RDF wordt in deze notitie gebruikt voor de combinatie van meerdere protocollen:
> RDF, RDFS en OWL 2.

Het GegevensWoordenboek Stedelijk Water (GWSW), een ontwikkeling van de stichting RIONED, is oorspronkelijk ontwikkeld in de Gellish taal. Ook Gellish is een semantische modelleringstaal in het zogenaamde ORO (Object-Relatie-Object) formaat. In het najaar van 2015 is het GWSW omgezet naar RDF.
In de “backend” situatie gebruikt het GWSW-projectteam het Gellish formaat nog steeds voor ontologie-ontwerp en -beheer. Via een geautomatiseerd proces wordt echter altijd gepubliceerd conform de RDF semantiek.

*Met dank aan:*
Daan Oostinga (Semmtech)
Mike Henrichs (Semmtech)
Michel Böhms (TNO)
Matthé van Koetsveld (CIM Architects)
Linda van den Brink (Geonovum)

## 1.1 Leeswijzer

Bij de uitwerking van deze tekst is er van uitgegaan dat de lezer bekend is met de principes en semantiek van RDF/RDFS/OWL 2 en het uitwisselformaat Turtle.
In de voorbeelden en in de praktijk (bij uitwisseling van GWSW-gegevens) gebruiken we het Turtle-formaat. Voor de concepten binnen de GWSW-Ontologie hanteren we in de voorbeelden de prefix “gwsw:”. Voor individuals in een dataset wordt de prefix “bim:” gebruikt.

In dit hoofdstuk vind u de begrippen en uitgangspunten bij de modellering in RDF. In het volgende hoofdstuk wordt samenvattend de opzet van het RDF model beschreven. In het laatste hoofdstuk vind u de gedetaillleerde uitwerking van het RDF model.

## 1.2 Gebruikte begrippen

RDF, RDFS, OWL 2
RDF staat voor Resource Description Format, de basisdefinitie van modellen op basis van subject-predicate-object. In de tekst verstaan we onder RDF de combinatie van RDF met RDFS (RDF Schema) en OWL 2 (Web Ontology Language).

Ontologie
Datastructuur die alle relevante entiteiten en hun onderlinge relaties en regels binnen een domein bevat. (bron: Wikipedia)

Individual
Een instantie van een concept, iets uit de werkelijkheid. Zoals individual “0980” de bestaande betonnen constructie van het soort/klasse/concept “rioolput” is.

Property
Voor de relatie (tussen subject en object) zijn meerdere namen gebruikelijk (“predicate”, “property name”), we hanteren in dit document “property”.

CE
De afkorting CE wordt gebruikt voor Class Expressions (in Description Logics “complex concepts”). CE’s worden ondermeer gevormd door Classes te binden aan Property Expressions. Met Class Expressions kunnen we onder andere restricties benoemen voor Properties waarmee concepten/klassen worden onderscheiden.

Dataset
Een dataset bevat de beschrijving van een fysiek stedelijk water systeem, de “individuals” op basis van het GWSW. De term ABox wordt ook gebruikt: “assertion components” binnen een ontologie. Voor het model (de concepten) wordt dan de term TBox gebruikt: “terminological components”.

## 1.3 Inrichting GWSW-OroX protocol

De GWSW ontologie wordt beschreven volgens het GWSW OroX protocol. Dit protocol onderscheidt zich door diepgang in semantiek en reikwijdte in de toepassing (van systeem tot proces). Het protocol beschrijft zowel het GWSW model (de "TBox", zie data.gwsw.nl) als de daarop gebaseerde datasets (de "ABox", zie apps.gwsw.nl).

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

- Soortenboom (de taxonomie of klasse-indeling)
- Samenstelling (de meronomie of deel-geheel indeling)
- Proces (het activiteiten-schema)

Belangrijke superklassen zijn:

- gwsw:FysiekObject
- gwsw:Activiteit
- gwsw:Ruimte
- gwsw:Levenvorm

Bij het ontwerp van een datastructuur spelen deze elementen de hoofdrol, ze vormen het ontwerpkader.
Met het principe van object-oriëntatie hanteert het model overerving-principes en maakt het zo expliciet 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 normalisatie-techniek om opslagruimte te beperken en redundantie te voorkomen.

## 1.4 Uitgangspunten

Voor de modellering is uitgegaan van het OWL RL (Rule Language) profiel. Dit profiel gebruikt nagenoeg alle OWL 2 semantiek en is toereikend voor het modelleren van de GWSW-Ontologie.

Voor de definitie van klassen, eigenschappen, datatypen en restricties kunnen verschillende benaderingen gekozen worden. De volgende uitgangspunten zijn gehanteerd:

- Bij het ontwerp van het GWSW in RDF wordt het oorspronkelijke Gellish-model zonder informatieverlies getransformeerd naar het RDF-model.
- In het verlengde daarvan: Voor de indeling in soorten, de vaststelling van de taxonomie, wordt de onderscheidende definitie zo expliciet mogelijk beschreven.
- Bij de modellering is rekening gehouden met de reasoner-prestaties, de benodigde rekenkracht voor de inferencing verschilt sterk per gekozen oplossing. Voor deze afweging is de Pellet reasoner versie 2.3.1 (voor OWL 2 RL) gebruikt.

## 1.5 Modelleerprincipes

Een groot deel van de gehanteerde modelleerprincipes stammen uit de oorspronkelijke opzet (gestart in 2006) van het model in Gellish-vorm. Deze principes zijn natuurlijk taalonafhankelijk, ook in de RDF-vorm blijven ze van groot belang. Veel dank gaat naar Andries van Renssen, geestelijk vader van Gellish en Matthé van Koetsveld, sterk betrokken bij de modellering in Gellish van het GWSW en zijn voorlopers.

### 1.5.1 Terminologie - Het vakgebied is leidend

(uitwerking: zie hst 3.1)

1. Volg de gebruikelijke termen binnen het vakgebied, bedenk geen nieuwe conceptnamen die misschien de lading beter dekken of neutraler zijn. Dat geldt ook - waar mogelijk - voor abstracte concepten.
2. Geef alle gebruikelijke vakgebied-termen die gelden voor het te modelleren systeem of proces een plek, als apart concept of als synoniem van een concept. De zoekfunctie moet volledig zijn.
3. Laat algemene termen die niet specifiek bij de discipline horen zoveel mogelijk buiten beschouwing. Modelleer bijvoorbeeld het concept "calamiteit" alleen als het als supertype nodig is.
4. Verwijs voor algemene termen waar mogelijk naar andere databronnen (rdfs:seeAlso).

### 1.5.2 Concepten en annotaties

1. Elk GWSW-concept is van het generieke type owl:Class.
2. Een concept wordt geïdentificeerd door de URI (prefix + naam) conform hst 3.1.1
3. Een concept wordt altijd voorzien van de annotaties zoals opgenomen in hst 3.1.2.

### 1.5.3 Specialiseren - Onderscheidende kenmerken

(het opbouwen van de soortenboom, uitwerking: zie hst 3.6)

1. Voor het classificeren van een concept uitgaan van zoeken op onderscheidende kenmerken in de (abstracte) soortenboom. Denk aan determineren van planten volgens Linnaeus: na het maken van een aantal keuzes wordt de soort gevonden
2. Streef ernaar om met de onderscheidende kenmerken de uitgeschreven definitie af te dekken
3. Gebruik de volgende onderscheidende kenmerken bij fysieke objecten:
    - Doel (waarvoor)
    - Toepassing (waarin)
    - Functie (wat doet het)
    - Uitvoering (hoe)
    - Structuur (waaruit)

4. Activiteiten worden altijd gekenmerkt door de relaties
    - Invoer / hasInput (het lijdend voorwerp)
    - Uitvoer / hasOutput (het gewijzigde voorwerp, een rapport)
5. Gebruik daarnaast de volgende onderscheidende kenmerken bij activiteiten:
    - Doel (waarvoor)
    - Toepassing (waarin)
    - Resultaat (wat doet het)
    - Technologie (werkwijze, eisen)
    - Mechanisme (waarmee)
6. Kwalificeer het onderscheidende kenmerk impliciet (gwsw:uitvoering "groot"). Expliciete kwalificaties (in de vorm van subtypes van generieke kenmerken) worden dus niet gebruikt (zijn - nog - onnodig).

### 1.5.4 Specialiseren - Abstracte concepten (en collecties)

1. Hou ze beperkt, de soortenboom zo veel mogelijk concreet houden. Supertypes zijn vaak alleen verdichtingen, geen soort.
2. Concepten zijn herkenbaar als abstract wanneer ze bijvoorbeeld niet in de deel-geheel relaties (bijvoorbeeld als deel van een rioleringsgebied) voorkomen.
3. Abstracte concepten bij voorkeur als groep/collectie (en niet als supertype) definiëren. Bijvoorbeeld groepering naar thema's, denk aan infiltratievoorzieningen.
4. Subtypes op hetzelfde niveau dienen in grote lijn hetzelfde samenstellingsniveau te hebben.

### 1.5.5 Specialiseren - Bladerobjecten

1. Specialiseer de concepten zoveel als mogelijk: definieer de subtypes, de "bladerobjecten".
2. Introduceer geen subtype als het geen onderscheidend kenmerk heeft. Bijvoorbeeld geen extra subtype "standaard hemelwaterstelsel" naast "verbeterd hemelwaterstelsel".
3. Hou er rekening mee dat de individuen zo specifiek mogelijk geclassificeerd dienen te worden. Classificatie met een supertype gebeurt alleen als het subtype niet van toepassing is (denk aan het eerdere voorbeeld "hemelwaterstelsel") of als het onbekend is en wel toegepast kan worden. Bijvoorbeeld bij gebruik van de inspectienorm voor "vrijverval rioolleidingen" (met subtypes gemengd, hemelwater, vuilwater).

### 1.5.6 Samenstellen - Definiërende onderdelen en verbindingen

1. Maak waar mogelijk de deel-geheel relatie definiërend. Een fysiek object heeft dan per definitie andere fysieke objecten als onderdeel.
2. Maak waar mogelijk ook de cardinaliteit definiërend, het aantal onderdelen is dan bepalend.

### 1.5.7 Samenstellen - Erven van samenstellingen

1. Voorkom redundantie van deel-geheel relaties, die relatie mag niet dubbel voorkomen voor een subtype en het bijbehorende supertype.
2. Definieer de samenstelling dus niet op een te hoog niveau (blijft verleidelijk)

### 1.5.8 Kenmerken - Horen exclusief bij een concept

1. Kenmerken zijn altijd eigendom van een entiteit/geheel en kunnen niet bestaan zonder de eigenaar.
2. Vermeng dus geen kenmerken van entiteiten. Bijvoorbeeld Dekselmateriaal, Beheerdernaam kunnen geen kenmerken van een Put zijn, die horen bij het concept Deksel en Beheerder. Die concepten zijn vervolgens gerelateerd aan de put.

3. Voorkom het opnemen van optionele kenmerken bij een supertype (kenmerken die niet voor alle subtypes gelden), definieer de kenmerken dus niet op een te hoog niveau.
4. Gebruik bijvoorbeeld het multi-parent principe. Als alleen kunststof leidingen het gebruikte kenmerk "kleur" hebben, introduceer dan het concept "kunststof leiding" met het kenmerk "kleur". Een indivual is dan zowel een "vrijverval rioolleiding" als een "kunststof leiding" zijn en heeft daarmee dat extra kenmerk.

### 1.5.9 Kenmerken - Geen typelijsten

1. Kenmerken die verwijzen naar een typelijst (bijvoorbeeld het kenmerk Soort Deksel van het concept Deksel) mogen niet voorkomen. Een typelijst moet altijd uitgedrukt worden in de taxonomie, bijvoorbeeld subtypes van Deksel.

### 1.5.10 Kenmerken - Specialiseren, maak ze intrinsiek

1. Specialiseer waar nodig de kenmerken, zodat restricties op de kenmerk-waarde zo specifiek mogelijk zijn. Bijvoorbeeld:
    - Definieer het generieke kenmerk "materiaal" met het subtype "materiaal put" met bijbehorende domeinwaarden
    - Definieer het generieke kenmerk "diameter" met het subtype "diameter leiding" met specifieke restricties op de afmetingen.
2. Intrinsieke kenmerken zijn geen noodzaak maar de combinatie met bepaalde restricties maakt ze waardevol en daarnaast wordt het model robuuster: als een individual het kenmerk heeft, dan hoort het van een bepaald type te zijn.

### 1.5.11 Contexten onderscheiden

TOP, BAS, hoe in te delen?
Opsplitsen in model-bestanden?
Combineren van model-bestanden in editor mogelijk? Onderscheid houden, ook na aanpassingen in de editor.
Enkelvoudige contexten altijd in apart model-bestand?

# 2 Samenvatting opzet GWSW-Ontologie in RDF

## 2.1 Explicite definitie: basis voor determinatie

Voor de indeling in soorten, de bepaling van de taxonomie, wordt de onderscheidende definitie zo expliciet mogelijk beschreven. Determinerend kan daarmee (de naam van) een soort worden bepaald. Verschillende elementen in de ontologie spelen hierbij een rol, die zijn beschreven in de volgende paragrafen.

### 2.1.1 Kwalificerende aspecten / onderscheidende kenmerken

De onderscheidende kenmerken specificeren de soorten, de GWSW ontologie hanteert de volgende:

- Doel (waarvoor)
- Toepassing (waarin)
- Functie (wat doet het)
- Uitvoering (hoe)
- Structuur (waaruit)

Meer specifiek voor activiteiten:

- Technologie (werkwijze, eisen)
- Resultaat
- Mechanisme (waarmee)

In RDF wordt het kwalificerend aspect beschreven door een CE met een restrictie op de property hasAspect gecombineerd met een restrictie op de property hasReference.

### 2.1.2 Kwalificerende samenstelling

De structuur wordt voor wat betreft de samenstelling expliciet beschreven door een CE met een restrictie op de property hasPart gecombineerd met de benoeming van de cardinaliteit. De cardinaliteit beschrijft het aantal voorkomens van een property tussen twee soorten.

### 2.1.3 Intrinsieke aspecten / possessed aspects

Afhankelijk van de soort kunnen kenmerken worden specialiseerd. Die intrinsieke kenmerken horen dan exclusief bij een soort. De CE beschrijft een restrictie op de property hasAspect in combinatie met het gerelateerd kenmerktype.

## 2.2 Metagegevens

Het oorspronkelijke Gellish-model bevat een serie metagegevens zoals Cardinaliteit Links/Rechts, UoM, Brondefinitie, Datum Begin/Wijziging. Die worden als volgt in het RDF-model meegenomen:

De Cardinaliteit wordt via CE's in RDF uitgedrukt. De cardinaliteit kan in twee richtingen gelden, daarvoor is voor de relevante properties een inverse geïntroduceerd. Ook voor deze omgekeerde propery geldt dan via de CE een restrictie op cardinaliteit.

De UoM, Brondefinitie en Datums worden als annotaties bij de concepten opgenomen.

De Taalgemeenschap wordt als extra naam bij de concepten (met property rdfs:label) vermeld.

De Collection Of Facts speelt een belangrijke rol in het GWSW- model, hiermee wordt aangegeven welk ORO-feit bij welke discipline/module (GWSW-Basis, GWSW-RIB, enz.) hoort. Binnen RDF wordt dit onderscheid gemaakt door aparte RDF-modellen per (combinatie van) van Collection Of Facts te maken. Het geldende "GWSW-filter" is altijd met de property rdfs:label in de ontologie opgenomen.

## 2.3 Top Level soorten

De "top level" concepten in GWSW-OroX zijn de concepten die boven in de soortenboom staan. Deze concepten zijn geen subklasse van andere concepten, ze zijn van het generieke type owl:Class.

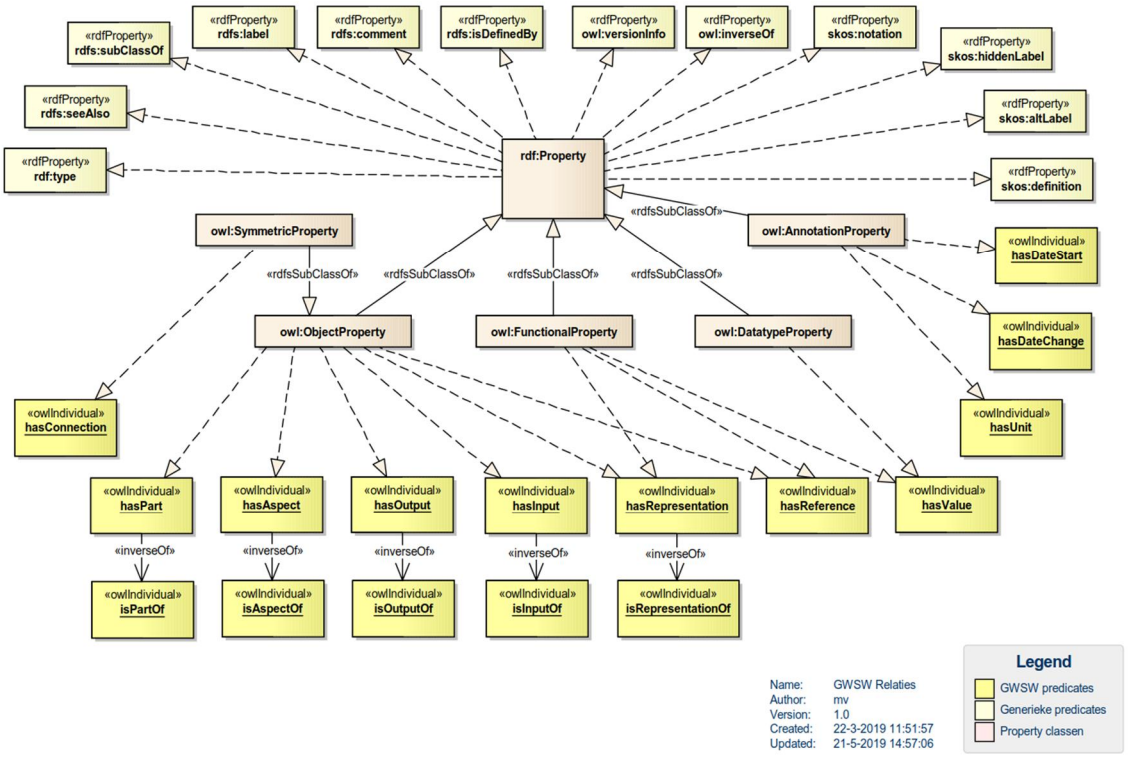
| Naam | Toelichting / definitie |
|---|---|
| Activiteit | [Gellish] Is a deed (a performance by changing a state of affairs) by carrying out an action or pattern of behavior. |
| FysiekObject | |
| Informatiedrager | [Gellish] Is a medium intended or actually carrying physical objects from that properties information can be deducted. Typically paper that carries ink, magnetic fields on a disk or tape, electrically or optically modulated substances of electricity or light. |
| Kenmerk | [Gellish] Is an individual object that is a phenomenon that is possessed by a totality and cannot exist without the existence of its possessor. It is an intrinsic, non-separable facet of its possessor |
| Levensvorm | Iets dat in staat is tot leven |
| Ruimte | [Gellish] Is an inanimate physical object (a physical object that is not capable of life) that is volumetric and is enclosed by physical or imaginary boundaries and exists in time. This is distinguished from a volumetric spatial aspect and from a volume as a property. |
| TopologischElement | [Gellish] Is a configuration that indicates how items are connected or positioned relative to each other. The topology in terms of connections usually doesn't change when the positions of the items change. |
| VerzamelingSoorten | Collection of qualitative aspects. Is specialisatie van collection of classes |
| | |
| | |

## 2.4 Collecties, domeintabellen

Alle collectie-leden zijn in de GWSW topologie opgenomen als individuen met annotaties. RDF beschrijft de enumeratie van individuals per collectie. Hoofdstuk 3.7 bevat de details.

## 2.5 Properties in de GWSW-Ontologie

De GWSW Relaties als diagram:




De tabellen beschrijven de opgenomen properties. De toepassing van properties (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 (“CE”).

| Property | Inverse property | Type | Omschrijving |
|---|---|---|---|
| owl:versionInfo | | | *Subject (ontologie)* heeft versieomschrijving *Literal* |
| rdf:type | | | *Subject* is van het type *Object* (Klasse-naam) |
| rdfs:subClassOf | | | *Subject* is van het subtype *Object* (Klasse-naam) |
| rdfs:label | | | *Subject* heeft als naam *Literal* (ook vertalingen, dan meerdere rdfs:label properties) (annotatie) |
| skos:altLabel | | | *Subject* heeft als synoniem *Literal* (ook vertalingen, dan meerdere rdfs:altLabel properties) (annotatie) |
| skos:hiddenLabel | | | *Subject* heeft als id *Literal* (annotatie, alleen gebruikt voor referentieobjecten, individuals zoals collectie-elementen) |
| skos:notation | | | *Subject* heeft als code *Literal* (annotatie) |
| skos:definition | | | *Subject* heeft als definitie *Literal* (definitie *zonder* bron-referentie) (annotatie) |
| rdfs:isDefinedBy | | | *Subject* is gedefinieerd door *Literal* (definitie *met* bron-referentie) (annotatie) |
| rdfs:seeAlso | | | *Subject* heeft aanvullende infomatie op *Literal* (URL) (annotatie) |
| rdfs:comment | | | *Subject* heeft als commentaar *Literal* (annotatie) |
| owl:inverseOf | | | *Subject-property* is de inverse van *Object-property* |
| hasUnit | | owl:AnnotationProperty | *Subject* heeft als eenheid *Literal* (annotatie) |
| hasDateStart | | owl:AnnotationProperty | *Subject* heeft als begindatum *Literal* (annotatie) |
| hasDateChange | | owl:AnnotationProperty | *Subject* heeft als wijzigingsdatum *Literal* (annotatie) |
| hasEditor | | owl:AnnotationProperty | *Subject* heeft als auteur *Literal* (naam persoon die concept heeft aangemaakt of gewijzigd) |
| hasFactColl | | owl:AnnotationProperty | *Subject* heeft als feitencollectie *Literal* (string met codes van één of meer collecties) |

| | | | |
|---|---|---|---|
| hasAspect | isAspectOf | owl:ObjectProperty | CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasAspect 0-n maal of min 0-n en max 1-n maal voorkomen |
| hasValue | | owl:DatatypeProperty<br>owl:FunctionalProperty | 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. |
| hasReference | | owl:ObjectProperty<br>owl:FunctionalProperty | CE beschrijft restrictie op object: Bij subject met property hasReference mogen alleen objecten van een bepaalde klasse (collectie) voorkomen |
| hasInput | isInputOf | owl:ObjectProperty | CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasInput 0-n maal of min 0-n en max 1-n maal voorkomen |
| hasOutput | isOutputOf | owl:ObjectProperty | CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasOutput 0-n maal of min 0-n en max 1-n maal voorkomen |
| hasPart | isPartOf | owl:ObjectProperty | CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasPart 0-n maal of min 0-n en max 1-n maal voorkomen |
| hasConnection | | owl:ObjectProperty<br>owl:SymmetricProperty | CE beschrijft restrictie op cardinaliteit: Bij subject mag property hasConnection 0-n maal of min 0-n en max 1-n maal voorkomen |
| hasRepresentation | | owl:ObjectProperty<br>owl:FunctionalProperty | |

Inverse properties zijn nodig om verschillen in cardinaliteit bij omgekeerde relaties te kunnen definiëren. Ze worden alleen gebruikt bij object-properties waarvan het type niet symmetrisch (hasConnection) of functioneel (hasRepresentation) is.

Voor het uitdrukken van CE's voorziet OWL 2 in een groot aantal (restrictie) properties. Daarmee kunnen we klassen expliciet 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 |

Hoofdstuk 3 beschrijft de toepassing van deze properties.

## 2.6 Properties in Datasets

In datasets conform het GWSW worden de volgende properties gebruikt:

| Property | Toelichting |
|---|---|
| rdf:type | *Subject* is van het type *Object* (klasse-naam) |
| rdfs:label | *Subject* heeft als naam *Literal* |
| hasAspect | *Subject* heeft als kenmerk *Object* |
| hasValue | *Subject* heeft als waarde *Literal* (subject is kenmerk) |

| | |
|---|---|
| hasReference | *Subject* heeft als referentie *Object* (subject is kenmerk) |
| hasInput | *Subject* heeft als invoer *Object* |
| hasOutput | *Subject* heeft als uitvoer *Object* |
| hasPart | *Subject* heeft als deel *Object* |
| hasConnection | *Subject* heeft verbinding met *Object* |
| hasRepresentation | *Subject* heeft als representatie *Object* |

## 2.7 Validatie/inferencing met GWSW-Ontologie

Hier volgt een opsomming van de mogelijke inferences en validaties. In enkele gevallen is reasoning op basis van het UNA (Unique Name Assumption) principe nodig. De controle op cardinaliteit is beperkt vanwege het OWA (Open World Assumption) principe in RDF.

- Controle op hasReference-waarden binnen domein van collecties / keuzelijsten (UNA)
- Controle op correcte typering binnen samenstellingen via "hasPart".
  - Ruimte hasPart "Object". Object: alleen van de klasse Ruimte of FysiekObject
  - FysiekObject hasPart "Object". Object: alleen van de klasse Ruimte of FysiekObject
- Inferencing: Individual-klasse wordt afgeleid uit intrinsiek aspect.
  - hasAspect BreedteLeiding => Individual = Leiding
- Inferencing: Individual-klasse wordt afgeleid uit onderscheidend kenmerk.
  - hasAspect Uitvoering + hasReference Klein => Individual = KleinObject
- Controle op correct gebruik datatype bij hasValue: decimal, string, integer, double, date, time, year.
- Controle op numerieke waarden binnen minimum maximum grenzen
- Cardinaliteit, aantal voorkomens per property boven het voor het type gedefinieerde maximum wordt gemeld (UNA)
  - ook "inverse"-cardinaliteit wordt in de reasoning meegenomen
  - minimum cardinaliteit en shall-relatie wel gemodelleerd, controle op strijdigheid met typering niet mogelijk (OWA)

# 3 Details van de GWSW semantiek

## 3.1 Concepten: naamgeving en annotaties

### 3.1.1 Naamgeving van concepten [URI]

Het hanteren van begrijpbare namen voor concepten is de gangbare RDF praktijk. We gaan uit van camelCase of CamelCase notatie van de namen voor respectievelijk de properties (starten met lowercase) als de klassen (starten met uppercase). De syntax van de namen is conform de voorwaarden voor een URI, de prefix + naam is de URI van het concept.
In het GWSW komen vertalingen voor, die worden als rdfs:label (voorzien van de taalcode) benoemd. Ook synoniemen komen veelvuldig in het GWSW voor, die worden benoemd met de relatie skos:altLabel.
Voor de property-namen wordt altijd de Engelse taal gebruikt.

In een dataset wordt via rdf:type verwezen naar de GWSW-conceptnaam:

```
bim:Put1        rdf:type        gwsw:Put .
```

Voorkeursterm GWSW-concepten
Voor de voorkeursterm van GWSW-concepten geldt het volgende uitgangspunt:
Altijd de literal bij rdfs:label als voorkeursterm gebruiken. Als die meertalig is (er kunnen meerdere rdfs:label relaties zijn met een eigen taalaanduiding) geldt altijd de voorkeur @nl en daarna @en (van een GWSW-concept is minimaal een @nl of een @en versie aanwezig).

In het oorspronkelijke Gellish-model is een nummer-identificatie (naast het unieke label) belangrijk. Dit unieke nummer wordt met de property skos:hiddenLabel benoemd, maar zal op termijn zijn waarde verliezen.

### 3.1.2 Annotaties bij concepten

De volgende annoties worden bij een GWSW-concept opgenomen:

| Annotatie | Omschrijving | Voorwaarden |
|---|---|---|
| rdfs:label | De voorkeursnaam van het concept | Exact 1 per taalgemeenschap. |
| skos:altLabel | Het synoniem van het concept | Onbeperkt (min=0) |
| skos:hiddenLabel | Het Gellish-ID, niet van toepassing voor nieuwe concepten buiten Gellish. De URI van het concept wordt de enige ID (hst 3.1.1) | Maximaal 1 (min=0) |
| skos:notation | De code van het concept, eventueel per context (zie verderop in dit hst) | Maximaal 1 per context (min=0) |
| skos:definition | Een "interne" omschrijving, vastgesteld binnen het GWSW-project | Onbeperkt (min=0) |
| rdfs:isDefinedBy | Een "externe" omschrijving | Onbeperkt (min=0)<br>Opbouw: [externe bron] Omschrijving |
| rdfs:seeAlso | Een verwijzing naar een externe bron | Onbeperkt (min=0)<br>Opbouw: URI (webadres) |
| rdfs:comment | Aanvullend commentaar | Onbeperkt (min=0)<br>Algemene opbouw: Commentaar-tekst<br>Verwijzing naar figuur: [Bijlage nnn.jpg]<br>- als "nnn" identiek is aan de URI-naam: [Bijlage *.jpg] |
| gwsw:hasUnit | Eenheid als string | Kies uit de tabel hierna |
| gwsw:hasDateStart | Datum waarop het concept is gemaakt | Exact 1 |
| gwsw:hasDateChange | Datum waarop het concept is gewijzigd | Maximaal 1, invullen als de waarde van één van de attributen wijzigt of als het concept andere properties (attributen/relaties) krijgt. Bevat altijd de laatste datum. |

| | | |
|---|---|---|
| gwsw:hasEditor (na versie 1.5.1) | Naam van de persoon die het concept voor het laatst heeft gewijzigd | Exact 1 |
| gwsw:hasFactColl (na versie 1.5.1) | Een literal met de verzameling codes van de factcollecties | Exact 1 |

Gebruik bij het attribuut hasUnit de tekst uit de kolom Eenheid:

| Eenheid | Datatype |
|---|---|
| % | xsd:integer |
| 1/h | xsd:decimal |
| 1/min | xsd:decimal |
| bar | xsd:decimal |
| degC | xsd:integer |
| dm3 | xsd:decimal |
| dm3/s | xsd:decimal |
| h | xsd:integer |
| m | xsd:decimal |
| m/dag | xsd:decimal |
| m2 | xsd:decimal |
| m3/h | xsd:decimal |
| mm | xsd:integer |
| mm/h | xsd:decimal |
| mm/min | xsd:decimal |
| yyyymmdd | xsd:date |
| yyyy | xsd:gYear |
| pcs | xsd:integer |
| ppm | xsd:integer |
| hhmmss | xsd:time |
| - *(factor)* | xsd:decimal |

Een voorbeeld van gebruikte annotaties:

```
gwsw:Put        rdf:type              owl:Class ;
                rdfs:label            "Put"@nl ;
                rdfs:subClassOf       :FysiekObject ;
                skos:definition       "Verticale waterdichte ….”@nl ;
                rdfs:isDefinedBy      "[IMGeo:1.0/2007] Gegraven of … "@nl ;
                rdfs:seeAlso          "http://www.w3.org" ;
                rdfs:comment          "Toelichting bij put" ;
                skos:notation         “AAA” .
```

Coderingen komen veel voor in het GWSW, bijvoorbeeld als taalonafhankelijke aanduidigen van toestandsaspecten in de EN13508-2. Codes van concepten zijn object bij de property skos:notation. In het voorbeeld is een fictieve code “AAA” gebruikt.
Er kunnen meerdere codes per context voorkomen. Voor het reiniging van een leiding worden bijvoorbeeld andere codes gebruikt dan voor het inspecteren van een leiding. Om dat onderscheid te kunnen maken is in de GWSW-Ontologie een datatype aan de code toegevoegd. Dat datatype representeert het geldende notatie-schema.

```
:StartNodeReference   skos:notation     “AAB"^^:Dt_Notation_RIB .     (inspecteren leiding)
:StartNodeReference   skos:notation     "GAB"^^:Dt_Notation_RRB .     (reinigen leiding)
:Dt_Notation_RRB      rdfs:label        "Codering reinigen put/leiding"@nl ;
                      rdf:type          rdfs:Datatype .
```

## 3.2 Kenmerken/aspecten – Class- of Property-central

### 3.2.1 Eigenschappen en predicaat

In RDF is het gebruikelijk om eigenschappen van een subject te benoemen in het predicate ("property-central"). Bijvoorbeeld:

```
bim:Put1        gwsw:hasAspectPutHoogte     1000^^xsd:integer .
```

Zo'n eigenschap/kenmerk kan ook als apart concept ("class-central") worden onderscheiden:

```
bim:Put1        gwsw:hasAspect              bim:Hgt1 .
bim:Hgt1        rdf:type                    gwsw:PutHoogte ;
                gwsw:hasValue               1000^^xsd:integer .
```

De notatie (in turtle) blijft overzichtelijk, het object Hgt1 kan anoniem blijven (zonder URI) en wordt bijvoorbeeld gecombineerd met de specificatie van putmateriaal (zie hoofdstuk "Collecties"):

```
bim:Put1        gwsw:hasAspect
                [
                  rdf:type                  gwsw:PutHoogte ;
                  gwsw:hasValue             1000^^xsd:integer
                ] ,
                [
                  rdf:type                  gwsw:PutMateriaal ;
                  gwsw:hasReference         gwsw:Beton .
                ]
```

In de GWSW Ontologie definieert voor veel kenmerken metagegevens zoals de "wijze van inwinning". Dat is volgens het class-central principe relatief eenvoudig te beschrijven:

```
bim:Put1        gwsw:hasAspect
                [
                  rdf:type                  gwsw:PutHoogte ;
                  gwsw:hasValue             1000^^xsd:integer ;
                  gwsw:hasAspect
                  [
                    rdf:type                gwsw:Inwinning ;
                    gwsw:hasAspect
                    [
                      rdf:type              gwsw:WijzeVanInwinning ;
                      gwsw:hasReference     gwsw:Schatting ;
                    ] ] ] .
```

Propery hasAspect is van het type owl:ObjectProperty.
Propery hasValue is van het type owl:DatatypeProperty en owl:FunctionalProperty. De property rdf:value wordt niet toegepast omdat het geen owl-type is (en daardoor bijvoorbeeld OWL RL-reasoning beperkt wordt).
Property hasReference is van het type owl:ObjectProperty en owl:FunctionalProperty.

In de GWSW ontologie gaan we uit van het "class-central" principe. Deze oplossing biedt een uitgebreidere semantiek en heeft (daarom) de voorkeur:

- Door objectificering van het kenmerk kan het in meerdere relaties gebruikt:
  - Het kenmerk kan met de relatie "hasInput" als object voor berekeningen staan
  - Het kenmerk kan zelf kenmerken bevatten (met de relatie "hasAspect"), bijvoorbeeld metagegevens over de inwinning. In het GWSW komt dit veel voor (zie het eerdere voorbeeld).

- Als alternatief voor de property-central oplossing met rdfs:range kunnen met een CE als subklassering op subject + property “hasValue” restricties aan het datatype worden toegevoegd. (zie hst. 3.2.2)
- Als alternatief voor de property-central oplossing met rdfs:domain kunnen met een CE als subklassering op het subject restricties aan de combinatie predicate en object worden toegevoegd. (zie hst. 3.2.3).
- In een dataset kan naar believen het aspect als URI of anoniem (via een blank node) worden uitgeschreven.
- De uitgebreidere semantiek kan in een dataset met beperkte syntax worden beschreven (zie het eerdere voorbeeld)

### 3.2.2 Minimum/maximum waarde, datatype

In het OroX worden twee soorten datatype gebruikt:

- de standaard xsd types: integer, decimal, date, gYear, time
- de OroX types: combinatie van datatype en waarde-restricties

Voor restricties op de kenmerk-waarde hanteren we:

```
gwsw:hasAspect      rdf:type                owl:ObjectProperty .
gwsw:hasValue       rdf:type                owl:DatatypeProperty ;
                    rdf:type                owl:FunctionalProperty . (waarde-relatie altijd max 1)
gwsw:PutHoogte      rdfs:subClassOf         gwsw:Kenmerk ;
                    rdfs:label              “Put hoogte” ;
                    rdfs:subClassOf
                    [
                      rdf:type              owl:Restriction ;
                      owl:onProperty        gwsw:hasValue ;
```

Alleen een restrictie op het standaard datatype:

```
                      owl:allValuesFrom     xsd:integer
                    ] .
```

Of restricties op min/max waarde met een GWSW-datatype:

```
                      owl:allValuesFrom     gwsw:dt_PutHoogte
                    ] .
gwsw:dt_PutHoogte   rdf:type                rdfs:Datatype ; (typering verplicht in OWL RL)
                    rdfs:label              “Put hoogte - datatype” ;
                    owl:equivalentClass
                    [
                      rdf:type              rdfs:Datatype ;
                      owl:onDatatype        xsd:integer
                      owl:withRestrictions
                      ( [xsd:minInclusive "0"^^xsd:integer] [xsd:maxExclusive    "10000"^^xsd:integer] )
                    ] .
```

### 3.2.3 Intrinsieke aspecten

Een intrinsiek aspect behoort specifiek (per definitie) bij een klasse.

```
gwsw:PutHoogte      rdfs:comment        “Intrinsiek kenmerk” ;
                    rdfs:subClassOf     gwsw:Hoogte .
```

Vanwege inference-snelheid hier eenzijdige subklassering aangehouden. Alleen de restrictie als subklasse is voldoende (“sufficient versus necessary”), type-prolongatie van CE naar Put. Wordt alleen (op deze manier) gebruikt in de dataset

Via blank-node subklasse bij Put met restrictie op property:

```
[                   rdf:type            owl:Restriction ;
                    owl:onProperty      gwsw:hasAspect ;
                    owl:someValuesFrom  gwsw:PutHoogte ;
]                   rdfs:subClassOf     gwsw:Put .
```

Met deze definitie worden Putten onderscheiden op basis van het intrinsieke aspect Puthoogte, de individual hoeft in de dataset niet getypeerd te worden:

```
bim:Put1            gwsw:hasAspect
[                   rdf:type            gwsw:PutHoogte ;
                    gwsw:hasValue       "1100"
] .
```

## 3.3 Kwalificaties

De restrictie-property owl:hasValue gebruiken we voor het kwalificeren van standaardwaardes van kenmerken (bijvoorbeeld voor versie-aanduidingen) .

```
gwsw:GwswVersie     rdf:type            owl:Class ;
                    rdfs:subClassOf
                    [
                     rdf:type           owl:Restriction ;
                     owl:onProperty     gwsw:hasReference ;
                     owl:hasValue       gwsw:Abc ;
                    ] .
gwsw:Abc            rdfs:label          “abc”^^xsd:string
                    rdf:type            gwsw:GwswVersie .      (wordt hiermee individual)
```

Met de restrictie-property owl:allValuesFrom worden concepten als kwalificatie benoemd.

```
gwsw:NodeId         rdf:type            owl:Class ;
                    rdfs:subClassOf
                    [
                     rdf:type           owl:Restriction ;
                     owl:onProperty     gwsw:hasReference ; (Objectproperty)
                     owl:allValuesFrom  gwsw:Rioolput
                    ] .
```

## 3.4 Samenstellingen (delen van, verbindingen)

De samenstelling-relaties komen voor in datasets en in de ontologie, in de ontologie worden alleen de restricties op deze relaties beschreven. De volgende relaties worden gebruikt:

- hasPart
- hasInput

hasOutput
hasConnection

Activiteiten
De relaties hasInput en hasOutput worden gebruikt voor de beschrijving van activiteiten en processen, een voorbeeld:

```
gwsw:hasInput          rdfs:label              "has as input" ;
                       rdf:type                owl:ObjectProperty .
gwsw:InspecterenPut    rdfs:label              "Inspecteren van een put"@nl, “Inspection manhole”@en ;
                       rdfs:subClassOf         gwsw:Activiteit .
```

In een dataset:

```
bim:Insp1              rdf:type                gwsw:InspecterenPut ;
                       gwsw:hasInput           bim:Put1 .
```

Verbindingen
hasConnection wordt van het type owl:SymmetricProperty.

Decompositie
hasPart wordt type owl:ObjectProperty en geldt voor zowel fysieke als ruimtelijke composities. Ruimtelijke composities (“bevatten” iets) via de property hasPart / isPartOf met restrictie op object-class:

```
gwsw:Ruimte            rdf:type                owl:Class ;
                       rdfs:label              "Ruimte"@nl ;
                       rdfs:subClassOf
                       [
                         rdf:type              owl:Restriction ;
                         owl:onProperty        gwsw:hasPart ;
                         owl:allValuesFrom
                        [
                          rdf:type             owl:Class;
                          owl:unionOf          (gwsw:Ruimte gwsw:FysiekObject) ;
                        ]
                       ] .
```

In combinatie daarmee is in de ontologie expliciet gemaakt dat bijvoorbeeld individuals van het type FysiekObject altijd iets anders zijn dan die van het type Ruimte:

```
gwsw:Ruimte            owl:disjointWith        gwsw:Kenmerk ;
                       owl:disjointWith        gwsw:FysiekObject . Enzovoort
```

## 3.5 Cardinaliteit

Cardinaliteiten worden in CE’s geborgd. Fictief voorbeeld van cardinaliteit voor objectproperty + objecttype:

```
gwsw:Rioolstelsel    rdfs:subClassOf
                     [
                       rdf:type                    owl:Restriction ;
                       owl:minQualifiedCardinality "1"^^xsd: nonNegativeInteger ;
                       owl:onProperty              gwsw:hasPart ;
                       owl:onClass                 gwsw:Lozingspunt .
                     ] ;
                     rdfs:subClassOf
                     [
                       rdf:type                    owl:Restriction ;
                       owl:maxQualifiedCardinality "99"^^xsd: nonNegativeInteger ; (onbeperkt = geen maximum)
                       owl:onProperty              gwsw:hasPart ;
                       owl:onClass                 gwsw:Lozingspunt .
                     ] .
```

Als de cardinaliteit niet beperkt is:

```
                     owl:minQualifiedCardinality      "0"^^xsd: nonNegativeInteger ;
```

Hiermee is wel gemarkeerd dat het aspect *relevant is voor* het subject.

Als de cardinaliteit verplicht voor een klasse:

```
                     owl:qualifiedCardinality         "1"^^xsd: nonNegativeInteger ;
```

Definiërende relaties
In het GWSW is beschreven welke relaties onderscheidend zijn voor de typering. Een rioolput moet bijvoorbeeld een deksel hebben om een echte rioolput te zijn. Daarvoor geldt ook een type-prolongatie van *Rioolput* naar CE, als iets een *Rioolput* is dan is het ook iets met een *Deksel*.

```
gwsw:Rioolput        rdfs:subClassOf
                     [
                       rdf:type                    owl:Restriction ;
                       owl:minQualifiedCardinality "1"^^xsd: nonNegativeInteger ;
                       owl:onProperty              gwsw:hasPart ;
                       owl:onClass                 gwsw:Deksel .
                     ] .
```

Inversed property
Cardinaliteit kan tweezijdig worden beschreven, daarvoor zijn er omgekeerde relaties nodig.

```
gwsw:isPartOf        rdfs:label                  "has as part (inverse)" ;
                     rdf:type                    owl:ObjectProperty ;
                     owl:inverseOf               gwsw:hasPart .
```

Ook voor deze inversed property + object (was subject) wordt dan de cardinaliteit gedefinieerd.

## 3.6 Onderscheidende kenmerken

Een onderscheidend kenmerk wordt gemodelleerd met restricties binnen een CE. Bij benoeming van de CE als equivalentClass vragen deze restricties veel rekenkracht, daarom is hier ook voor een éénzijdige subtypering (rdfs:subClassOf ipv owl:equivalentClass) gekozen. Daarmee leveren we wel semantiek in (“sufficient”, niet “necessary”).

Definieer Onderscheidend Kenmerk

```
gwsw:Uitvoering      rdfs:comment                "Onderscheidend kenmerk"@nl ;
                     rdfs:subClassOf             gwsw:Kenmerk .
```

Combineer het kenmerk en de waarde ervan in een CE

```
gwsw:KleinObject    rdfs:subClassOf          gwsw:FysiekObject .
[                   rdf:type                 owl:Restriction ;  Via blank node: eenzijdige subklasse
                    owl:onProperty           gwsw:hasAspect ;
                    owl:someValuesFrom
                    [
                     owl:intersectionOf      zowel klasse Uitvoering als CE-restrictie op hasReference
                     ( gwsw:Uitvoering
                     [
                       rdf:type              owl:Restriction ;
                      owl:onProperty         gwsw:hasReference ;
                      owl:hasValue           gwsw:Klein;  (individual)
                     ] ) ]
]                   rdfs:subClassOf          gwsw:KleinObject .
gwsw:Klein          rdfs:label               "klein" ;
                    rdf:type                 gwsw:Uitvoering . (wordt hiermee individual)
```

Als in de dataset een invidual als volgt beschreven is kan een reasoner op basis van de ontologie afleiden dat KleinObject1 van het type KleinObject is:

```
bim:KleinObject1    gwsw:hasAspect
                    [
                     rdf:type               gwsw:Uitvoering ;
                     gwsw:hasReference      gwsw:Klein .
                    ]
```

## 3.7 Collecties

Voor de modellering van collecties gebruiken we in RDF een enumeratie van individuals. Alle collectiemembers (elementen) zijn dus in de GWSW-topologie opgenomen als individuen met annotatieproperties.

```
gwsw:PutMateriaal   rdfs:subClassOf          gwsw:Kenmerk ;
                    rdfs:label               "Put materiaal" ;
                    rdfs:subClassOf
                    [
                     rdf:type               owl:Restriction ;
                     owl:onProperty         gwsw:hasReference ;  (FunctionalProperty + ObjectProperty)
                     owl:allValuesFrom      gwsw:PutMatColl ;
                    ] .
gwsw:PutMatColl     rdf:type                 owl:Class ;
                    rdfs:subClassOf
                    [
                      rdf:type              owl:Class ;
                      owl:oneOf             (gwsw:Beton gwsw:Pvc)  (individuals)
                    ] .
gwsw:Beton          rdfs:label               "beton" ;              (annotatie: naam)
                    rdf:type                 gwsw:PutMatColl ;      (algemene typering)
                    skos:hiddenLabel         111004801 ;            (nummer-id van concept)
                    skos:notation            "A" .                  (annotatie: code)
```

In de dataset verwijzen naar het individu:

```
bim:Put1        rdf:type            gwsw:Put ;
                gwsw:hasAspect
                [
                rdf:type            gwsw:Put_mat ;
                gwsw:hasReference   gwsw:Beton ;        (Objectproperty)
                ] .
```