INFORMATIE VLAANDEREN

/// HANDLEIDING

OSLO MODELLERINGSREGELS

Versie /// 0.7

Publicatiedatum /// x/x/2019

Vlaamse overheid

www.vlaanderen.be/informatievlaanderen

Auteur: Geert Thijs

Datum aanmaak: [datum]

Datum afdruk: 14 maart 2019

Interne bestandsnaam: OSLO Modelleringsregels11.docx

Documenthistoriek:

Versie	Opmerking	Datum	Auteur	Status
0.1	Inhoudstafel uitgewerkt obv referenties ISO, OMG, W3C. Algemene regels deels uitgewerkt.	11/6/2018	Geert Thijs	Draft
0.2	Structuur herwerkt, inleiding en aanpak verder uitgewerkt. Aanzet model- en schemaregels.	21/6/2018	Geert Thijs	Draft
0.3	Model- en schemaregels verder uitgewerkt.	17/8/2018	Geert Thijs	Draft
0.4	Typo's verbeterd. Model-en schemaregels verder uitgewerkt, sommige herwerkt. Figuren aangepast: UpperCamelCase voor associaties. Layoutregels uitgewerkt.	4/9/2018	Geert Thijs	Draft
0.5	Diagram metamodel aangepast. Annex met UML-profiel toegevoegd. Begin verwerking input AWV. Aparte paragraaf voor standaardenregister.	26/11/2018	Geert Thijs	Draft
0.6	Verdere verwerking input AWV. Overblijvende TODO's verwerkt. Toegevoegde paragrafen: Uitbreiding, Pakketten, Zichtbaarheid, Documentatie, Inhoud vh model, Specialisatie van structurele kenmerken. Afgesplitste paragrafen: Overerving, Aggregaat/composiet.	31/01/2019	Geert Thijs	Draft
0.7	Typo's verbeterd. Aangepaste paragrafen: Subklassen of enumeratie.	x/x/2019	Geert Thijs	Draft

Informatie Vlaanderen
Havenlaan 88, 1000 Brussel
+32 (0)2 553 72 02
Koningin Maria Hendrikaplein 70, 9000 Gent
+32 (0)9 276 15 00
informatie.vlaanderen@vlaanderen.be

INHOUD

1 INLEIDING

OSLO ontwikkelt semantische datastandaarden, ttz datastandaarden die semantische interoperabiliteit bevorderen in een bepaald domein. Dat domein wordt voorgesteld door een model dat de concepten in het domein en de relaties ertussen definieert.

Deze aanpak vormt de basis van het semantisch web (een web van data ipv louter van documenten), dat ervan uitgaat dat niet enkel de data zelf maar ook de betekenis ervan moet worden uitgewisseld om voldoende interoperabiliteit tussen applicaties te bekomen.

OSLO volgt deze principes en ontwikkelt voor diverse domeinen zgn vocabularia en applicatieprofielen, resp modellen van een domein en van toepassingen in dat domein. OSLO maakt geen implementatiemodellen.

Hoewel de manier om modellen formeel voor te stellen anders is bij datastandaarden voor het semantisch web, verschillen de regels niet van deze bij klassieke modellen. Het gaat steeds om:

- Regels die betrekking hebben op het modelleren zelf, ttz het herleiden van een domein tot objecten, attributen etc.
- Regels voor het opstellen ve schema, di de formele voorstelling van het model (bv dmv UML-klassen en een klassendiagram).

OSLO combineert klassieke modelleringsregels met regels die specifiek zijn voor het semantisch web. Deze handleiding documenteert deze regels.

2 AANPAK

In dit deel beschrijven we de algemene aanpak die OSLO volgt bij het modelleren. Om duidelijk te maken waar die vandaan komt beschrijven we eerst de modelleringsaanpak van de ISO TC211, de OGM en het W3C Semantisch Web.

Tegelijk wordt daardoor een aantal termen mbt modellering verduidelijkt.

2.1 ISO TC211

De ISO TC211 werkgroep (werkgroep voor geografische standaarden) ontwikkelt standaarden die een bepaald domein (Universe Of Discourse) beschrijven. De abstractie ve dergelijk domein noemen we een model.

ISO (ISO 19103, 2015), p50 maakt gewag van volgende soorten modellen:

- Conceptual model
- Specification model (= applicatiemodel), waarvan twee soorten worden onderscheiden:
  - Information model (= domeinmodel, zie (ISO 19103, 2015), p51)
  - Service model
- Implementation model

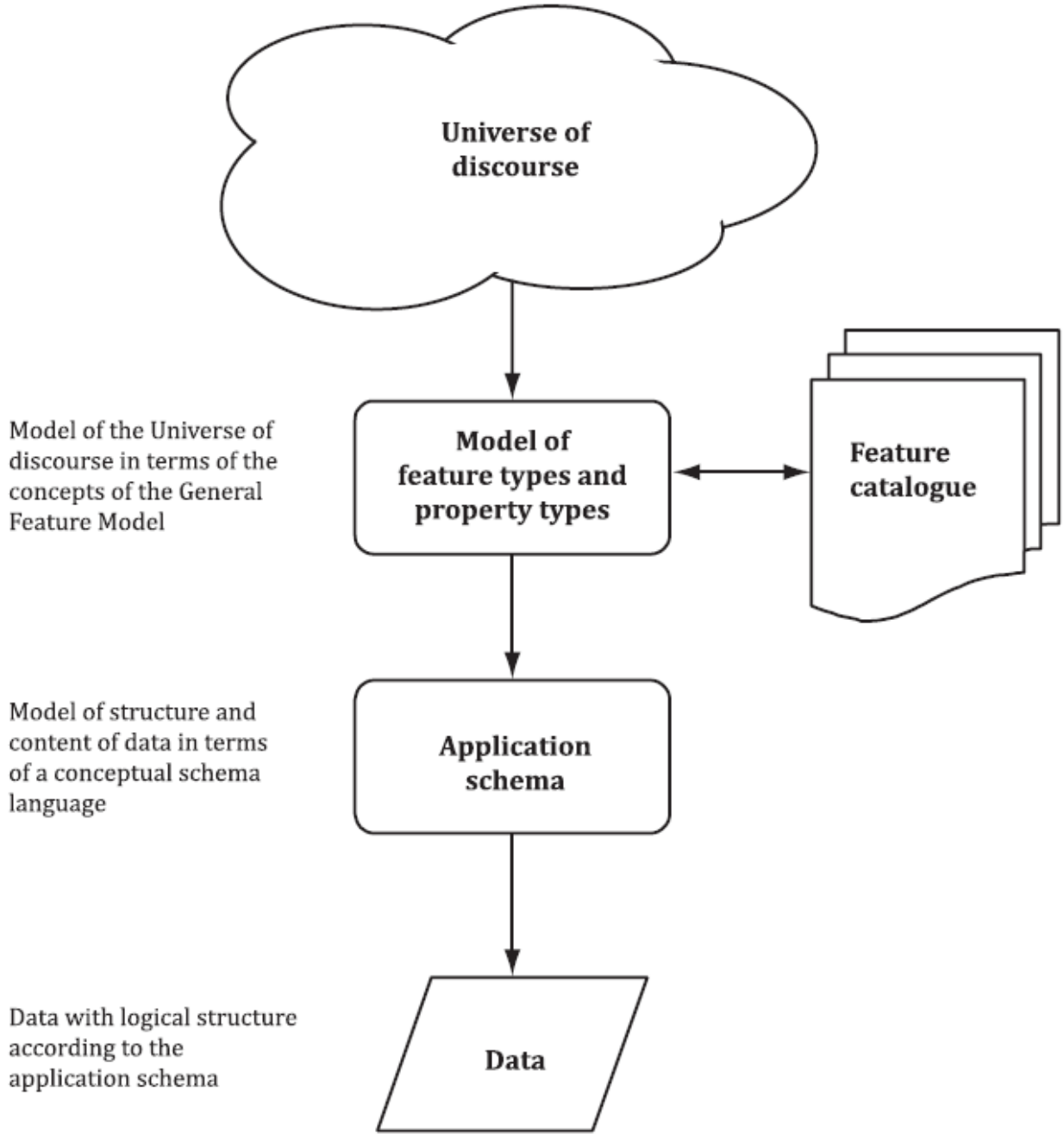
Conceptual model & Specification model worden samengevoegd in één model.

Het modelleren gebeurt tov een ISO TC211 metamodel, (het zgn GFM, zie §3.2.1), een model bestaat steeds uit elementen uit dat metamodel: FeatureTypes, AttributeTypes, AssociationTypes etc.

Het bekomen model wordt beschreven door een objectcataloog (ISO 19110) en op een meer formele manier door een zgn schema (ISO 19109, 2015). Daarbij gebruikt ISO TC211 een UML-klassendiagram dat dmv een UML-profiel beperkt werd tot de elementen uit het metamodel.

Machineleesbare uitwisseling vh schema gebeurt dmv het UML-uitwisselingsformaat XMI.

Deze figuur vat de ISO TC211 terminologie mbt modellering samen:



Bron: (ISO 19109, 2015), p10.

2.2 OMG

De OMG onderscheidt volgende modellen (OMG, 2014):

- Computation Independent Model (CIM)
- Platform Independent Model (PIM)
- Platform Specific Model (PSM)

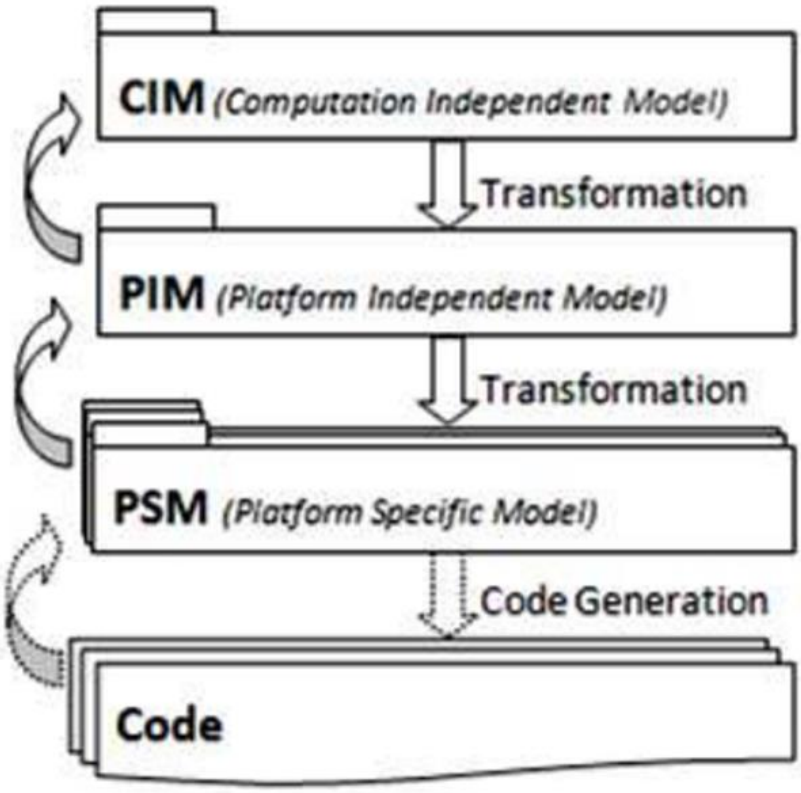
Dit past binnen de terminologie van Model Driven Architecture (MDA) vh OMG waarbij in een aantal stappen uit een conceptueel model code voor een specifiek platform wordt afgeleid. Idee hierachter is “separation of concerns”: het loskoppelen van business overwegingen van de details vd technische implementatie.

Het CIM is het conceptueel model: een opsomming van de concepten in een bepaald domein en de use cases vh te ontwikkelen systeem.

Het PIM is een formele representatie vh CIM, standaard een UML-klassendiagram.

Een PSM is een PIM dat zodanig is gemanipuleerd (typisch door stereotyping, dmv een UML-profiel) dat er voor een bepaald technologisch platform (SQL, XML, RDF) code uit kan worden afgeleid (bv een DDL-script om een databank op te bouwen, een XSD om een SOAP-service te beschrijven, een contextfile voor JSON-LD…).

Geïllustreerd als volgt:



Bron: (Kriouile)

Het modelleren gebeurt volgens het UML-metamodel (zie §3.2.1), ttz een model zal bestaan uit Klassen, Attributen, Associaties etc. Een formele voorstelling ve model gebeurt dmv een UML-klassendiagram. Machineleesbare uitwisseling gebeurt dmv het UML-uitwisselingsformaat XMI.

Ter info: UML & XMI zijn ISO-standaarden ontwikkeld door de OMG. Zie (OMG, 2015) voor de UML-specificatie.

2.3 W3C SEMANTISCH WEB

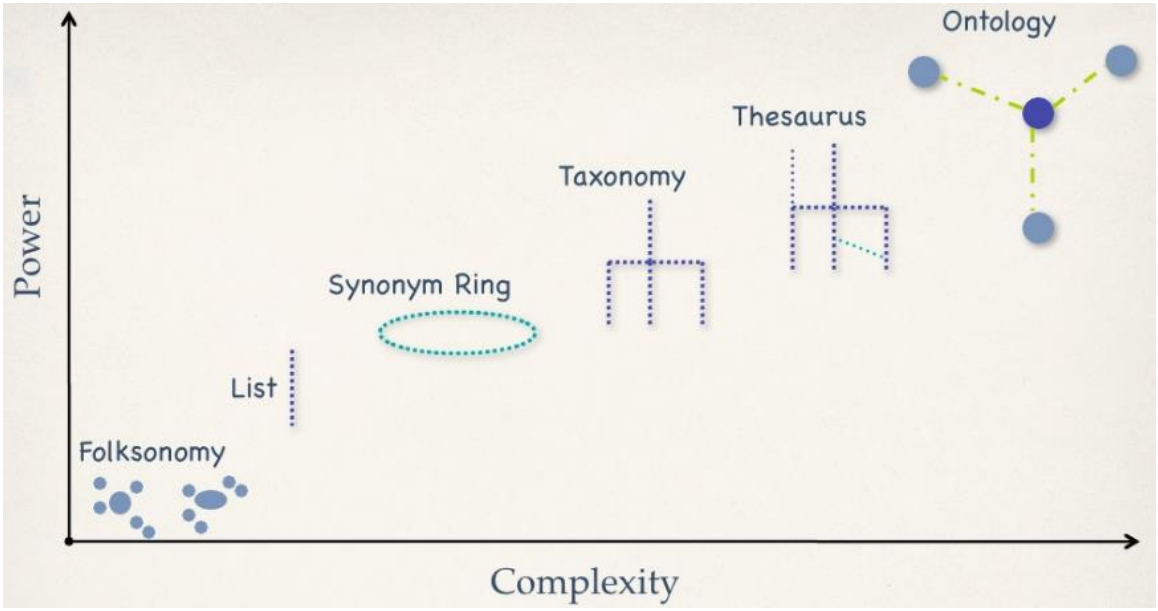
W3C ontplooit initiatieven om een web van data te ontwikkelen ter vervanging van het huidige web van documenten. Technologieën als RDF om data te linken, SPARQL om data te bevragen en RDFS/OWL om data te modelleren zijn daarvoor ontwikkeld. Hier zijn we vooral geïnteresseerd in dat laatste en W3C maakt daarbij een onderscheid tussen een:

- Vocabulary (VOC)
- Application Profile (AP)

Een vocabularium (in zijn meest complexe vorm ook wel ontologie genoemd) definieert de concepten en de relaties tussen concepten (ook wel termen genoemd) in een bepaald domein.

Opmerking: Kenmerkend is dat een VOC uitbreidbaar is. Het Semantisch Web beschouwt het als open, ttz iedereen kan termen toevoegen die hij/zij noodzakelijk acht. Men baseert zich op de Open World Assumption (OWA) die stelt dat een uitspraak waar is zolang met niet weet of ze onwaar is. Niemand heeft volledige kennis: het is niet omdat een attribuut niet voorkomt dat het geen kenmerk is vd beschouwde klasse, het ontbrekend attribuut kan altijd later nog aan het model worden toegevoegd.

Er bestaat een spectrum van soorten vocabularia waarbij complexiteit en mogelijkheden telkens toenemen:



Bron: (Connors, 2009)

Een ontologie is de meest complexe vorm en wordt gemodelleerd volgens het metamodel van RDFS en/of OWL (zie §3.2.1), ttz een ontologie bestaat uit elementen zoals Classes, Properties etc. De minder complexe vormen van vocabularia (bv codelijsten) worden beschreven dmv SKOS.

Het begrip “applicatieprofiel” wordt niet expliciet gedefinieerd door W3C, maar is een algemeen gangbare term in de informatica die we hier gebruiken om aan te geven dat ihkv een applicatie of klasse van applicaties een selectie van concepten is gemaakt uit de vocabularia en dat constraints zoals kardinaliteiten en codelijsten zijn toegevoegd.

Opmerking: Een AP is gesloten, uitbreiding kan enkel in een volgende versie van het AP of buiten het AP (bv door het AP te importeren in een nieuw AP). Een AP steunt op de Closed World Assumption (CWS), het tegengestelde vd Open World Assumption vermeld hierboven. CWS veronderstelt volledige kennis, alle noodzakelijke attributen zijn aanwezig.

De formele manier om een VOC te beschrijven is RDFS/OWL, op het laagste niveau (dat vd RDF waarmee RDFS/OWL wordt beschreven) voor te stellen dmv een graph. In praktijk echter wordt meestal een UML-klassendiagram gebruikt, al dan niet volgens het UML-profiel ODM (zie §3.2.1) dat UML beperkt tot de elementen die in het RDFS/OWL metamodel voorkomen.

Er is officieel geen formele manier om een applicatieprofiel te beschrijven, maar zowel OWL als SHACL worden als oplossing genoemd.

Machineleesbare uitwisseling gebeurt in RDF.

2.4 OSLO

OSLO gebruikt de principes en technologie vh semantisch web, maar steunt daarnaast op de klassieke modelleringsregels van het ISO TC211 en de OMG.

Volgende termen mappen daarbij op elkaar:

- Vocabularium: Komt overeen met het conceptueel model zoals de ISO het bedoelt en met het CIM vd OMG. Verschil is dat een VOC uitbreidbaar is.
- Applicatieprofiel: Is gelijk aan het informatiemodel (domeinmodel) of servicemodel vd ISO en het PIM vd OMG.

Het semantisch web doet niet aan implementatiemodellen en ook OSLO niet: de modelleringsregels van OSLO hebben louter betrekking op conceptuele modellen en op applicatiemodellen.

Bovendien hebben de modelleringsregels momenteel enkel betrekking op informatiemodellen (domeinmodellen), niet op servicemodellen.

Het metamodel waarop OSLO zich baseert om een domein te modelleren is dat van de UML-klassen. Dwz dat OSLO-modellen bestaan uit Klassen, Attributen, Associaties etc, zie §3.2.1 voor meer info.

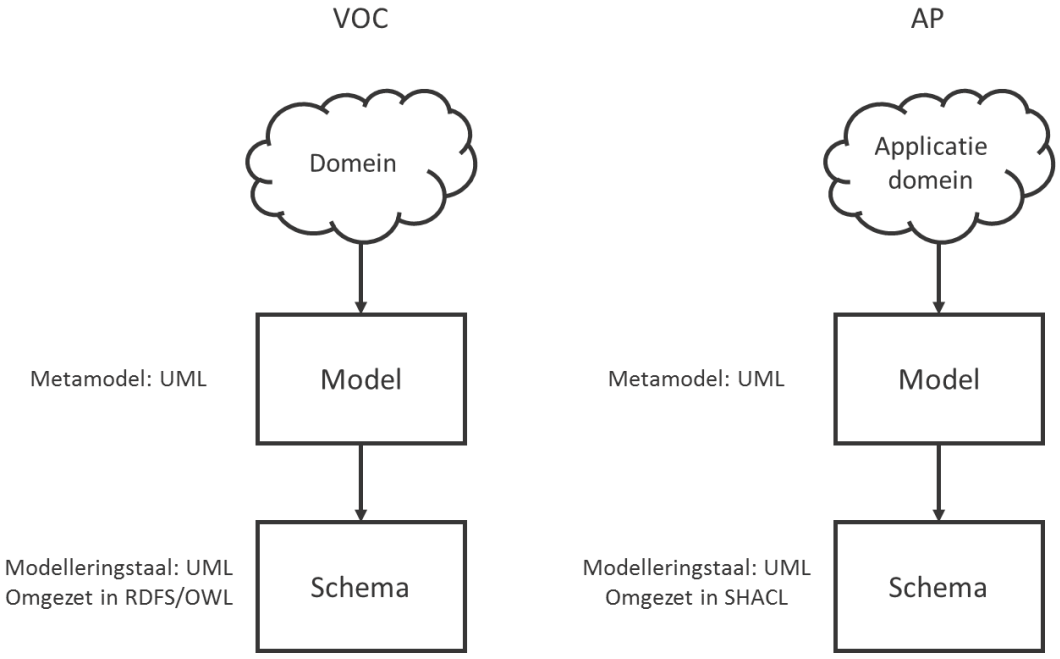
Opmerking: Dit is meestal ook de aanpak op het semantisch web. In een model kunnen daardoor wel elementen voorkomen die in het RDFS/OWL metamodel niet gekend zijn, bv associaties.

De formele manier om het bekomen model te beschrijven (het zgn schema) is een UML-klassendiagram, voor een VOC daarna omgezet in RDFS/OWL en voor een AP in SHACL. Die omzettingen maken vh CIM of PIM (dat met resp het VOC of AP overeenstemt) een PSM, specifiek voor het RDF-platform.

Opmerking: De omzetting betekent dat sommige UML-elementen uit het oorspronkelijk model vertaald moeten worden naar overeenkomstige elementen in het RDFS/OWL-model. Bv een UML-Associatie wordt omgezet in een RDFS/OWL Property.

Opmerking2: De omzetting naar RDFS/OWL & SHACL wordt toegelicht in een ander document.

Het machineleesbaar uitwisselingsformaat voor het UML model is XMI, voor RDFS/OWL & SHACL is dat RDF. Een codelijst wordt beschouwd als een minder complexe ontologie en wordt bij OSLO beschreven dmv SKOS. Volgende figuur illustreert de OSLO-modelleringsaanpak:



3 REGELS

We onderscheiden volgende soorten regels:

1. Algemene regels
2. Model- en schemaregels
3. Regels mbt diagramlayout

Opmerking: De voorbeelden die worden gegeven bij de regels zijn zoveel mogelijk ontleend aan bestaande OSLO-standaarden.

Opmerking: De tool waarmee de illustraties werden vervaardigd is Enterprise Architect van SPARX.

3.1 ALGEMENE REGELS

Het betreft regels over (zie ook (OGC)):

- Inhoud vh model.
- Hergebruik.
- Modulair ontwerp.
- Scope.
- Abstractieniveau.
- Versionering.
- Afhankelijkheden.
- Schema.
- Standaardenregister.
- Uitbreiding.
- Documentatie.

3.1.1 Inhoud vh model

Zoals vermeld in §2.4 omvat een VOC/AP:

- Een model
- Een schema

Het model is de collectie van concepten die het domein en/of de applicatie omschrijft, het schema is de meer formele representatie vh model. Voor beide gebruiken we UML 2.5 als modelleringstaal.

De abstracte syntax van UML omvat de elementen vh UML metamodel voor klassen, ttz Klassen, Attributen, Associaties etc. Zie §3.2.1 voor meer info.

De concrete syntax van UML is grafisch: de meer formele weergave vh model gebeurt dmv een UML-klassendiagram, zie §3.1.8.

De elementen ve model worden gegroepeerd in een Pakket, meer daarover in §3.2.23.

Een VOC bevat de concepten nodig om een bepaald domein te beschrijven (bv het domein van Adressen), een AP de concepten nodig om een applicatie in een bepaald domein te beschrijven (bv een Adresregister).

Een AP brengt concepten uit meerdere VOC's samen, het legt in feite uit hoe die concepten gebruikt moeten worden voor een bepaalde toepassing. Niet alle concepten uit de gebruikte VOC's zijn daarbij nodig, een AP omvat typisch een subset van de beschikbare concepten.

Omdat een AP bruikbaar moet zijn voor specifieke toepassingen worden er bijkomende eisen aan gesteld:

- De multipliciteit van elementen wordt toegevoegd, multipliciteit komt in VOC's niet voor (zie §3.2.14).
- Aan de elementen in het model worden eventueel constraints opgelegd, in een VOC komt dit niet voor (zie §3.2.16).
- Enumeraties worden uitgewerkt, in een VOC wordt gezegd dat het datatype een enumeratie is maar niet welke enumeratie (zie §3.2.17).
- Definities worden verfijnd tov deze in een VOC (zie §3.2.21).

3.1.2 Hergebruik

Hergebruik van bestaande concepten is een voorwaarde voor semantische interoperabiliteit.

REGEL: Hergebruik maximaal bestaande concepten.

3.1.2.1 Vormen van hergebruik

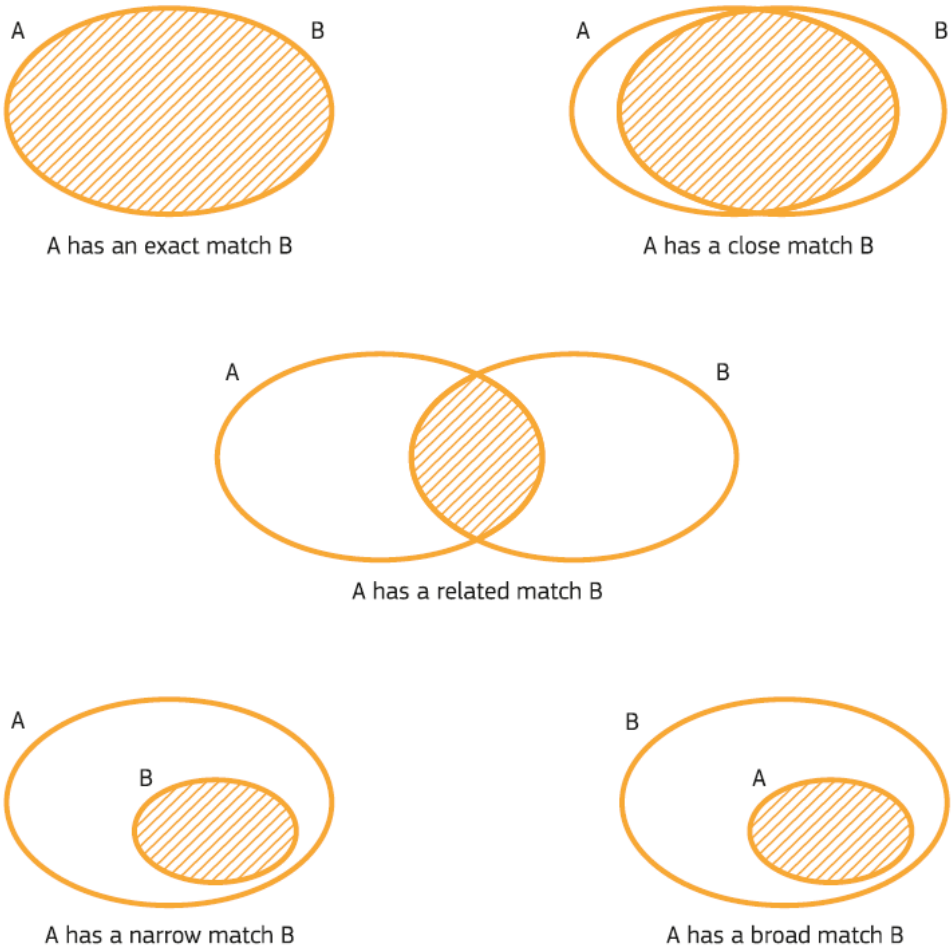
Hergebruik kan door:

- Importeren
- Aligneren
- Relateren
- Specialiseren

REGEL: Herdefinieer geen bestaand concept.

Bv als een bestaand VOC al een klasse Persoon bevat, creëer dan zelf geen klasse Persoon (met dezelfde naam en/of dezelfde definitie). Bestaande concepten kunnen wel worden uitgebreid, zie §3.1.10.

Een bruikbare tool om uit te drukken in welke mate een concept overeenstemt met een bestaand concept zijn de SKOS-mapping categorieën, zie (W3C):



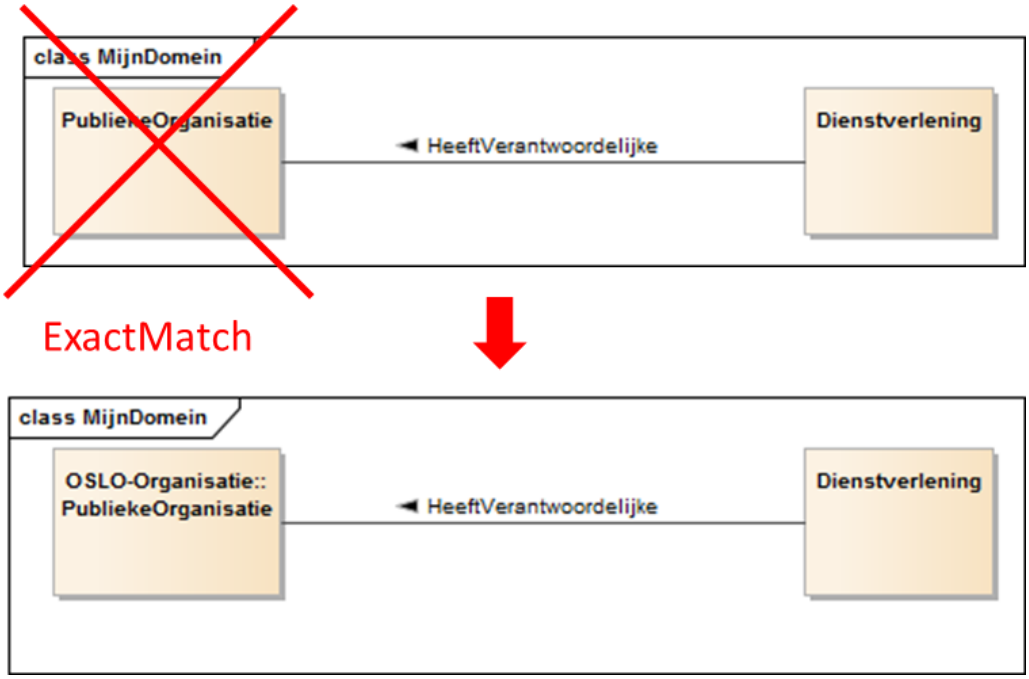
Bron: (ISA)

Het resultaat vd mapping bepaalt welke vorm van hergebruik het meest geschikt is.

3.1.2.1.1 Importeren

REGEL: Als het concept een ExactMatch heeft met een bestaand concept, importeer dat concept dan in het model.

Vb een AP Dienstverlening importeert het concept PubliekeOrganisatie uit een VOC Organisatie:



3.1.2.1.2 Aligneren

REGEL: In het geval van een CloseMatch pas indien mogelijk de definitie vh eigen concept aan.

Vergelijk bv twee onderstaande definities vh concept Gebouw:

DefinitieOSLO	DefinitieINSPIRE
Een gesloten en/of overdekt, bovengronds of ondergronds bouwwerk, dat dient of bestemd is, ofwel om mensen, dieren en voorwerpen onder te brengen, ofwel om economische goederen te vervaardigen of diensten te verstrekken. Een gebouw verwijst naar gelijk welke structuur die op blijvende wijze op een terrein opgetrokken of gebouwd wordt.	A Building is an enclosed construction above and/or underground, used or intended for the shelter of humans, animals or things or for the production of economic goods. A building refers to any structure permanently constructed or erected on its site.

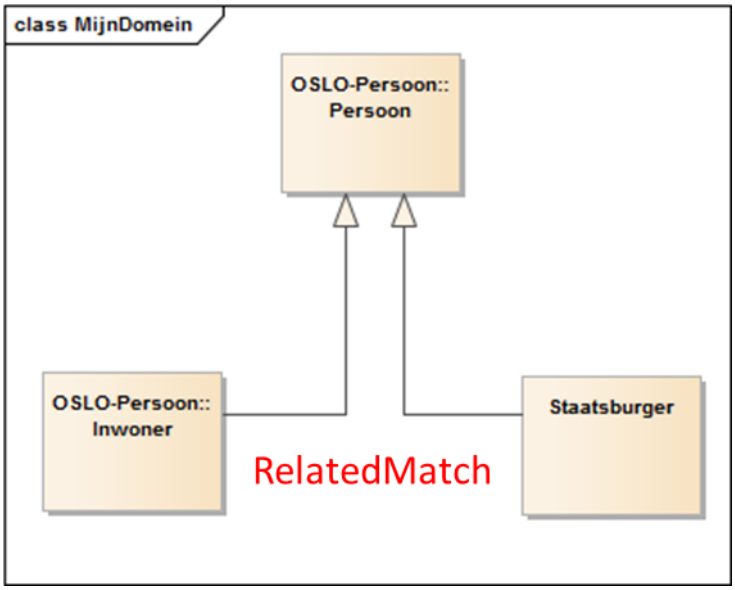
CloseMatch

Opmerking: Dit kan impact hebben op de inwinningscriteria, bv Gebouw = gesloten constructie vs Gebouw = gesloten of overdekte constructie.

3.1.2.1.3 Relateren

REGEL: In het geval van een RelatedMatch voeg het nieuwe concept toe en materialiseer indien mogelijk de relatie.

In dit vb gebeurt dit door de verwante begrippen Inwoner en Staatsburger neer te zetten als specialisatie van Persoon:



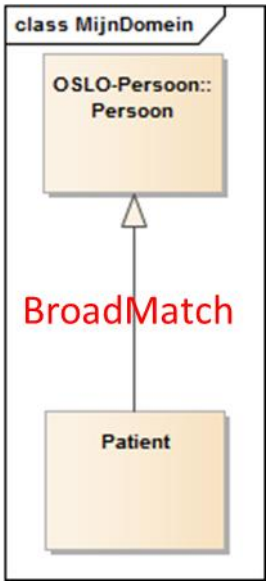
In dit vb voegen we een nieuw concept Vestiging toe en associëren dat met het verwant concept Organisatie:

Opmerking: De toegevoegde relaties kunnen beschouwd worden als uitbreidingen vh bestaande concept (en dus vh bestaande VOC). Zie §3.1.10 voor meer info.

3.1.2.1.4 Specialiseren

REGEL: Indien het concept een BroadMatch heeft met het bestaand concept, specialiseer dan het bestaand concept.

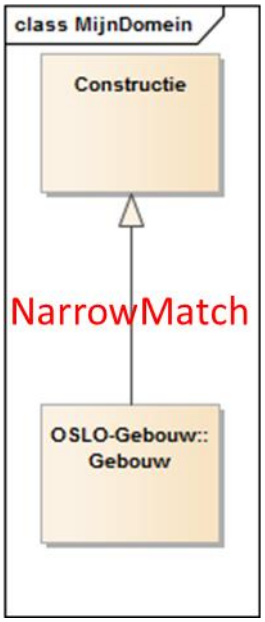
Vb:



Hoewel minder gebruikelijk kan het omgekeerde ook:

REGEL: Indien het concept een NarrowMatch heeft met het bestaand concept, generaliseer dan het bestaand concept.

Vb:



Opmerking: De toegevoegde specialisaties/generalisaties kunnen beschouwd worden als uitbreidingen vh bestaande concept (en dus vh bestaande VOC). Zie §3.1.10 voor meer info.

3.1.2.2 Herkomst vd concepten

VOC's en AP's verwijzen naar elkaar:

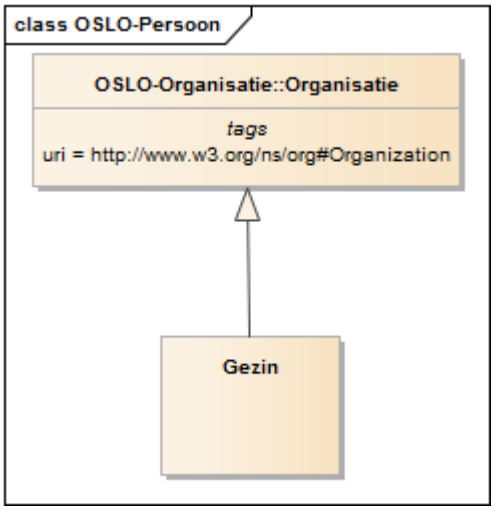
- VOC herbruikt elementen uit bestaand VOC
- AP herbruikt elementen uit bestaand VOC
- AP herbruikt elementen uit bestaand AP

3.1.2.2.1 VOC herbruikt elementen uit bestaand VOC

REGEL: VOC herbruikt VOC: Herbruik in volgorde van voorkeur:

1. Concept uit OSLO-VOC.
2. Concept uit datastandaard op het semantisch web (zgn mapping).
3. Concept uit andere datastandaard.

Gebruik bij voorkeur concepten uit OSLO-VOC's omdat deze VOC's een samenhangend geheel vormen en maximaal concepten hergebruiken. In een OSLO-VOC komen zowel nieuwe als bestaande concepten voor, bv in het VOC OSLO-Persoon:



Uitleg bij dit vb: Nieuw concept in OSLO-Persoon is Gezin, Organisatie bestaat al ih VOC OSLO-Organisatie en is op zijn beurt gemapt op het concept Organization in het W3C ORG Vocabulary.

Een overzicht vd huidige OSLO-VOC's is te vinden in het OSLO-Standaardenregister, meer info in §3.1.9.

Komt een concept niet voor in het OSLO-VOC (noch als nieuw concept, noch als bestaand concept), ga er dan zelf naar op zoek.

Geef daarbij de voorkeur aan VOC's gepubliceerd op het semantisch web. Deze concepten hebben een eigen uri waardoor het model/schema er ondubbelzinnig naar kan verwijzen (dmv een uri-tag, zie §3.2.19 over metadata). Het geheel van dergelijke verwijzingen in een VOC noemen we de mapping (omdat ze in principe het resultaat is van de mapping-oefening vermeld in §3.1.2.1).

Vb van mapping: zie bovenstaand vb waar Organisatie gemapt werd op Organization. Nog een ander vb, eveneens uit een OSLO VOC:

Is een bestaand concept niet gepubliceerd op het web dan kan men het nog steeds gebruiken maar is het niet mogelijk om er formeel naar te verwijzen. Verwijs in dat geval naar de datastandaard in de samenvatting bij de specificatie.

Om na te gaan of een bepaald domein al door een bestaand VOC wordt beschreven zie oa het LOV directory en het JOINUP platform.

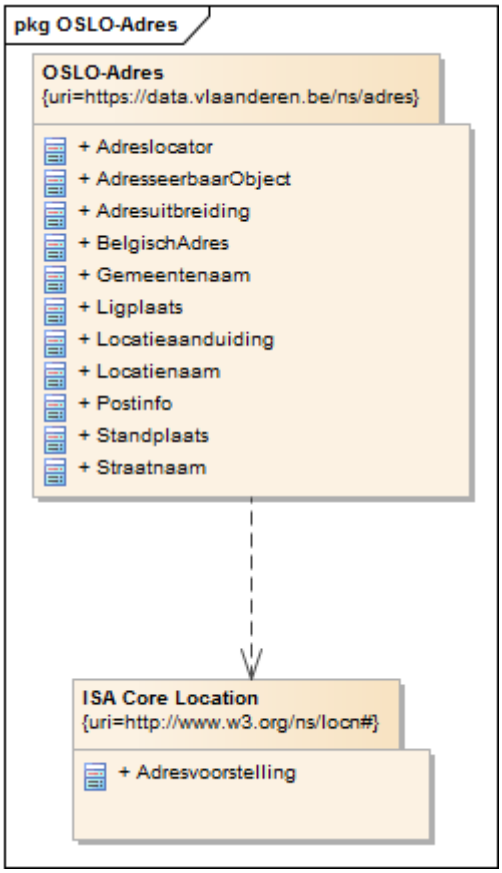
Voor regels om de geschiktheid ve gevonden VOC te beoordelen zie §6 in (W3C, 2014).

3.1.2.2.2 AP herbruikt elementen uit bestaand VOC

REGEL: Baseer een AP steeds op bestaande VOC's.

Dit volgt uit de definitie ve AP: een selectie van concepten uit één of meer VOC's voor een bepaald soort toepassingen, zie §2.3 voor de volledige definitie.

Creëer desnoods een nieuw OSLO-VOC of voeg concepten toe aan bestaande VOC's om de regel te realiseren. Bv voor een AP OSLO-Adresregister was het ISA Core Location Vocabulary te beperkt en bleek een VOC OSLO-Adres noodzakelijk:



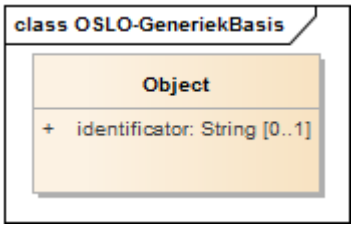
Opmerking: Ook voor het uitbreiden ve bestaand VOC moet een nieuw OSLO-VOC worden gemaakt, tenzij men eigenaar is vh VOC, zie §3.1.10.

Los van de herkomst vd concepten die erin worden gebruikt voegt een AP nog zaken toe zoals multipliciteiten, specifieke enumeraties, aangepaste definities (zie §3.1.1). Die zijn eigen ah AP, bv eenzelfde concept kan in verschillende AP's een andere multipliciteit hebben.

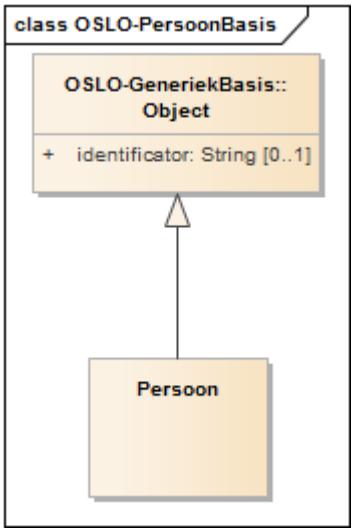
3.1.2.2.3 AP herbruikt elementen uit bestaand AP

Herbruik in dat geval enkel de concepten die het AP uit diverse VOC's selecteerde en pas de geldende beperkingen toe zoals multipliciteit, dezelfde enumeraties etc.

Bv als in OSLO-GeneriekBasis geldt:



Dan blijft bij overname door OSLO-PersoonBasis de multipliciteit van identificator dezelfde:



Hergebruik ve AP door een ander AP is mogelijk bij OSLO maar verschijnt niet automatisch in de documentatie, vandaar volgende regel:

REGEL: AP herbruikt AP: Vermeld welk AP werd herbruikt.

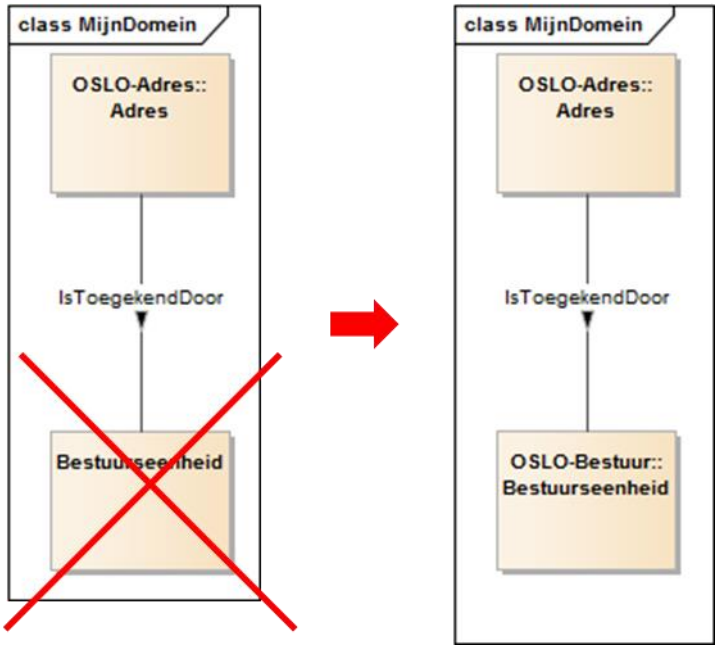
Bv door de namespace te tonen zoals hierboven, zie ook §**Error! Reference source not found.** over afhankelijkheden.

3.1.3 Modulair ontwerp

REGEL: Als een concept bruikbaar is voor meerdere domeinen, plaats het dan in een apart domein.

Opmerking: Domein wordt hier gebruikt in de betekenis van Universe of Discourse.

Vb: bestuurseenheden zijn bruikbaar in meerdere domeinen en dus creëren we er een apart domein voor:



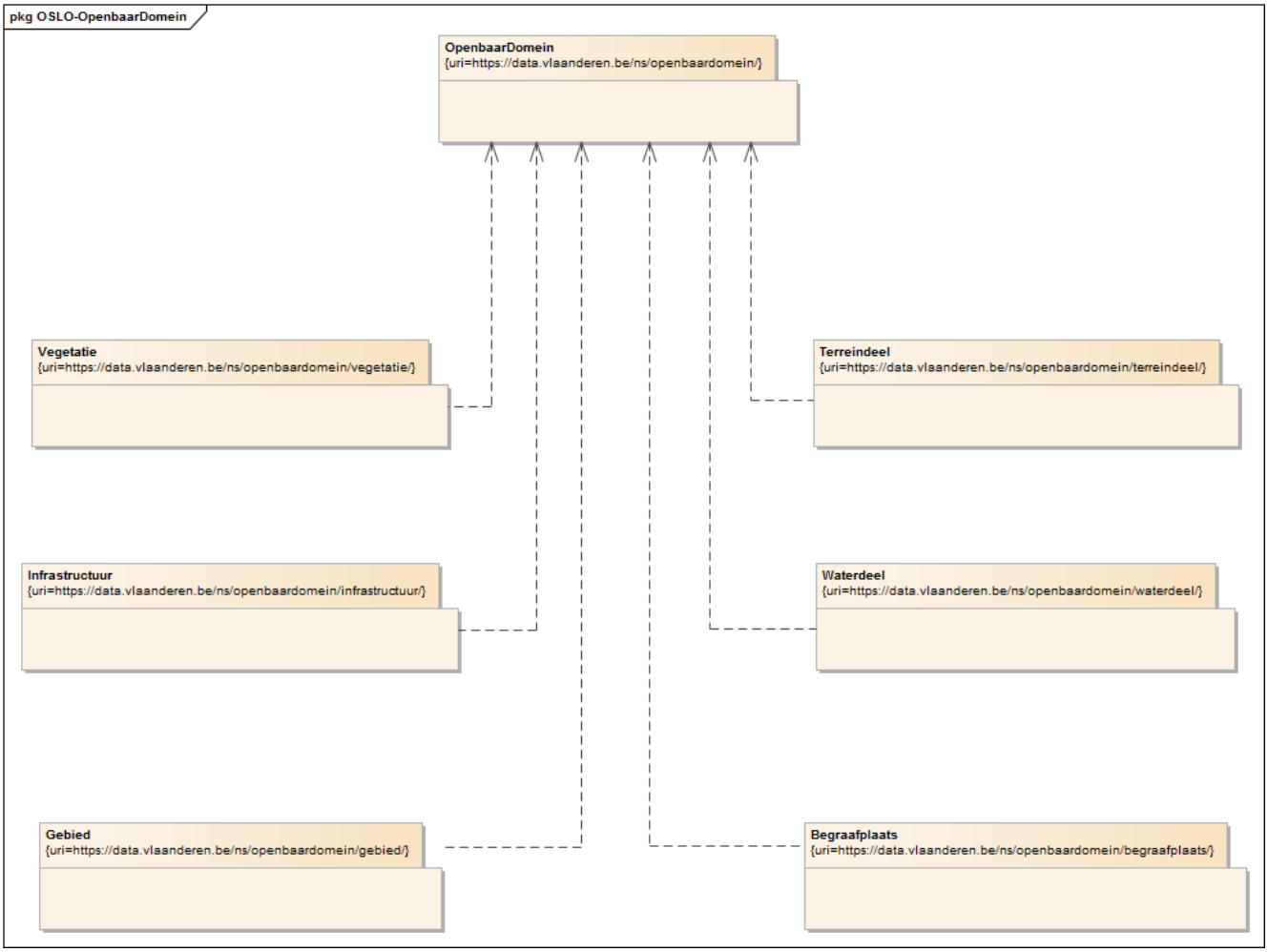
REGEL: Vermijdt circulaire referenties.

Dwz: model1 verwijst naar model2 verwijst naar model1.

En verder:

REGEL: Als een domein erg veel concepten bevat, splits het dan op in subdomeinen.

Vb:



Bv het domein Infrastructuur is een uitbreiding vh domein OpenbaarDomein.

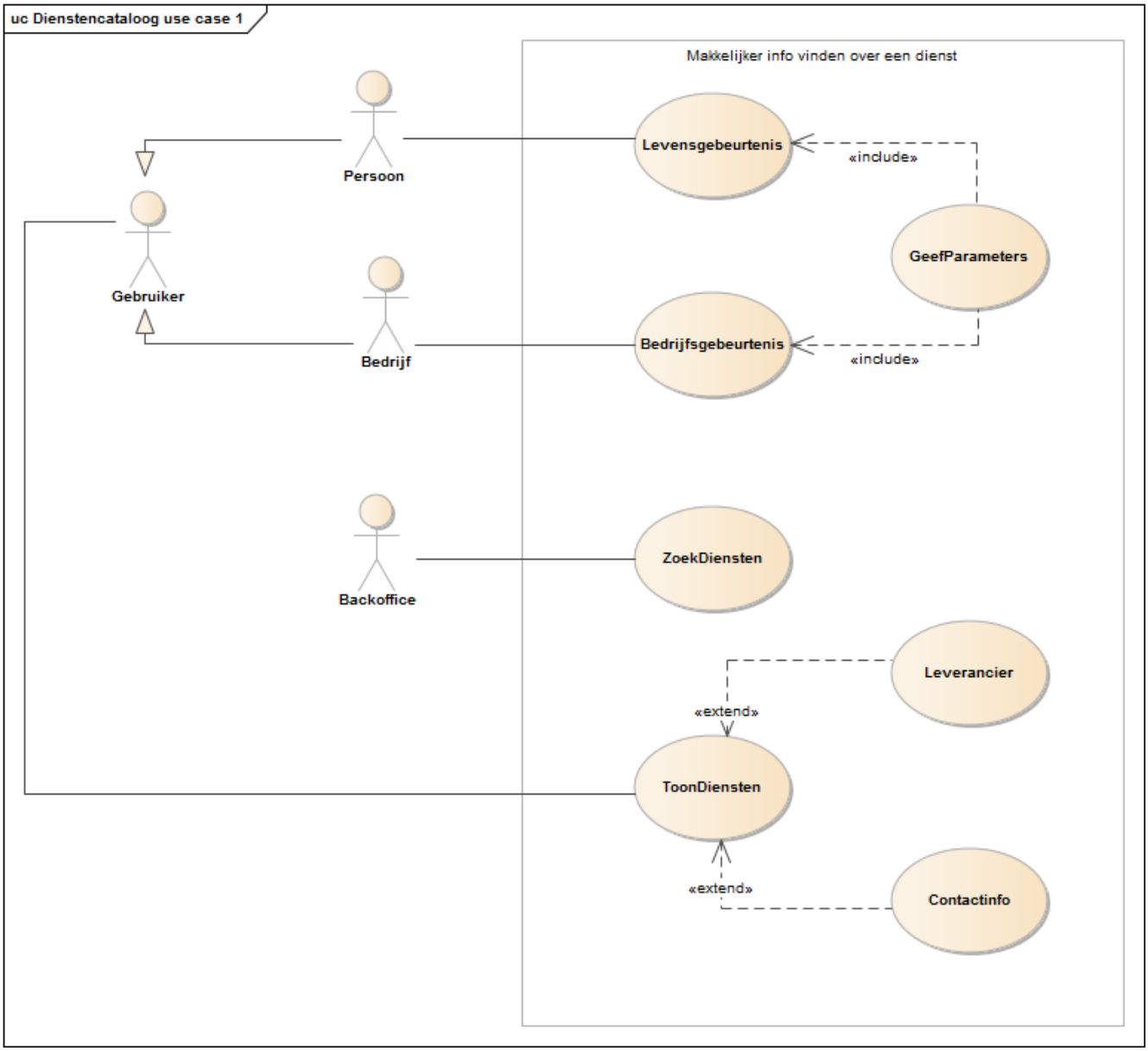
Opmerking: We gaan er bij OSLO van uit dat de begrippen domein/model/pakket doorgaans samenvallen. Zie §3.2.23.

3.1.4 Scope

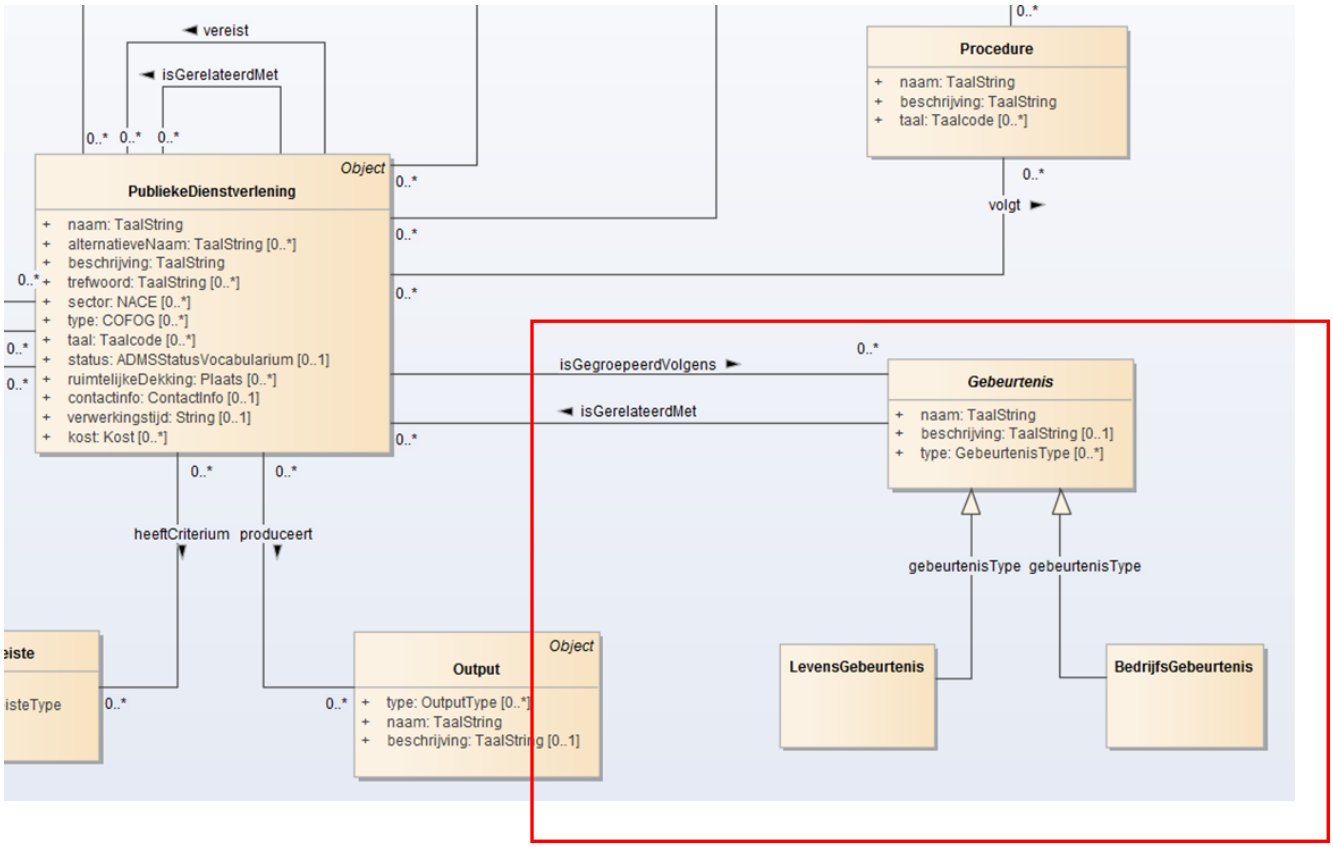
REGEL: Bepaal de scope van een model adhv de use cases waarvoor het wordt opgesteld.

Dit geldt voor de AP's aangezien die een bepaald soort applicatie voor ogen hebben. De scope van een VOC is uitgebreider, het is de optelsom van de scopes van alle AP's die er gebruik van maken.

Vb van een use case voor een AP Dienstencataloog:



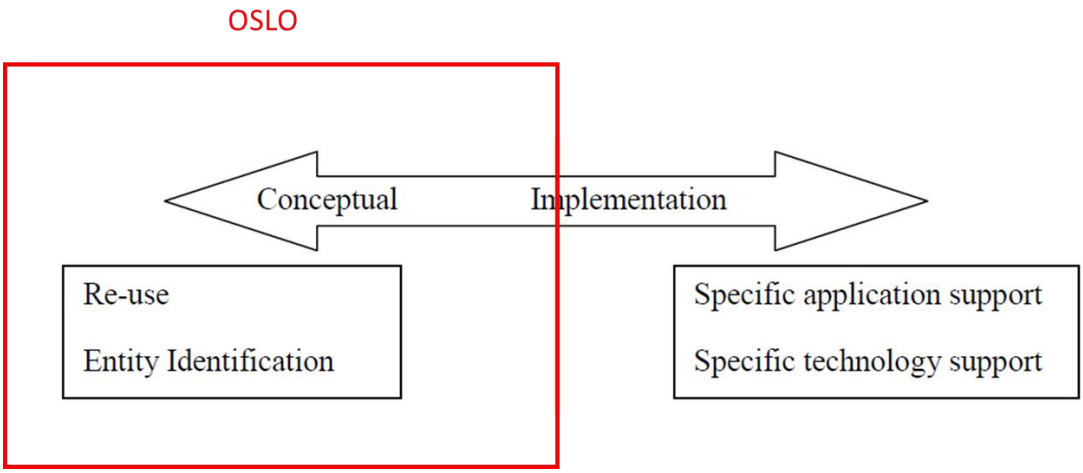
Uitleg bij dit vb: Een use case van een Dienstencataloog is het vinden ve dienstverlening. Bedoeling is dat de gebruiker de dienst zoekt niet op basis vd organisaties die deze mogelijk aanbieden maar op basis van waarvoor hij/zij de dienst nodig heeft. Maw: niet zoeken op BurgerlijkeStand maar op GeboorteAangeven. Dit is bepalend voor het AP in die zin dat daarin het concept Persoonsgebeurtenis/Bedrijfsgebeurtenis moet voorkomen en dat de diensten volgens deze gebeurtenissen gegroepeerd moeten worden:



3.1.5 Abstractieniveau

REGEL: Ontwikkel modellen die noch platform-specifiek, noch implementatiemodellen zijn.

Deze figuur geeft weer waar de grens ligt:



Bron: (OGC)

De generieke use cases voor OSLO zijn:

- Uitwisseling (exploratief of formeel)
- Hergebruik bij het opstellen van nieuwe modellen
- Mapping (transformatie) tbv integratie & publicatie

Aan deze use cases wordt niet voldaan als:

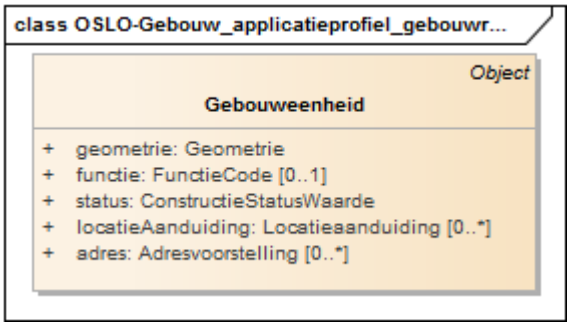
- Het model een specifieke applicatie beschrijft (vb GRB, GIPOD, AR/GR…)
- Het model een specifiek platform ondersteunt (bv XSD, DDL, RDF)

Maw: overspecificatie ondermijnt semantische interoperabiliteit.

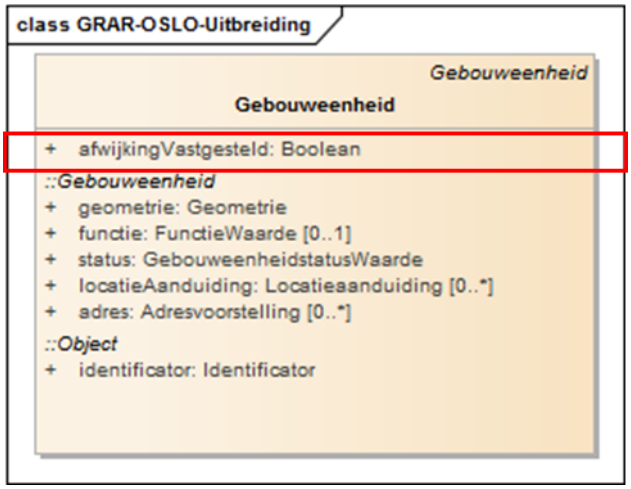
Concreet komt dit erop neer dat er geen attributen voorkomen zoals een id of metadata. Ook wordt zo min mogelijk technische terminologie gebruikt.

Voorbeeld specifieke applicatie:

AP OSLO-Gebouwregister



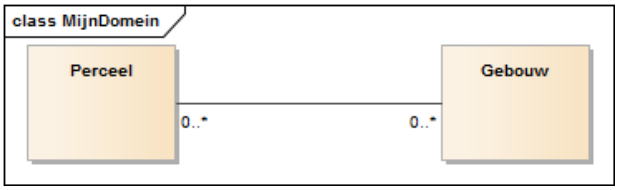
GebouwRegister-AdresRegister



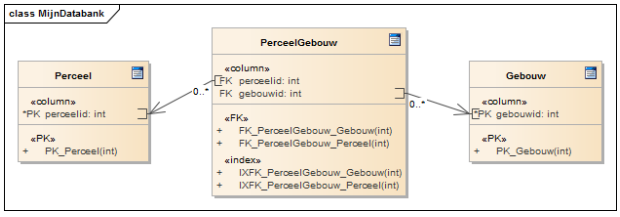
Uitleg bij dit vb: Het attribuut Gebouweenheid.afwijkingVastgesteld geeft aan of de Gebouweenheid al dan niet voldoet aan de inwinningscriteria. Dit is relevant voor het GebouwRegister-AdresRegister maar niet voor een standaard Gebouwregister omdat een register in principe enkel geldige objecten bevat.

Voorbeeld specifiek platform:

AP OSLO-Gebouwregister



DDL Gebouwregister



Uitleg bij dit vb: Het rechtermodel is een implementatie in een RDBMS vh applicatiemodel links. Typisch worden in een relationele databank veel-veel-relaties voorgesteld door een tabel en worden de tabellen aaneen gekoppeld dmv sleutelvelden. Dergelijke technische elementen horen echter niet thuis in een OSLO-model, hetzelfde model op een ander platform zou er immers helemaal anders kunnen uitzien.

Een concept dat in dit verband extra aandacht verdient is tijd. Hiermee bedoelen we niet zozeer temporele datatypes (moment, interval, periode) of soorten tijd (geldige tijd, transactietijd of user-defined tijd) maar wel het type uitspraken dat men bij het modelleren over tijd kan doen, ttz uitspraken over de toestand ve object ih domein:

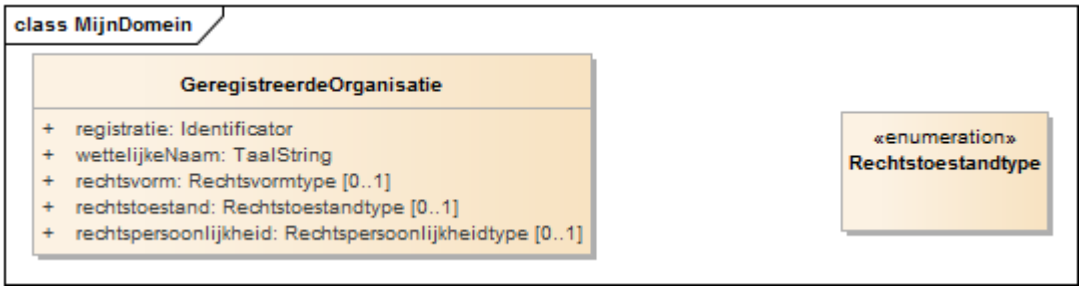
- De huidige toestand (current)
- De opeenvolging van toestanden (sequenced)
- De voorgekomen toestanden (nonsequenced)

Meer hierover in (Snodgrass, 2000). Punt is dat hier volgende regel uit volgt:

REGEL: Beperk het model tot concepten die de huidige toestand van objecten ih domein beschrijven.

De mogelijkheid om de historiek ve gegeven te kunnen uitwisselen beschouwen we als een vereiste die door de applicatie wordt bepaald en waarvan de uitwerking afhangt vh platform. Vanuit semantisch oogpunt volstaat het om de actuele toestand vd objecten ih domein te kunnen beschrijven.

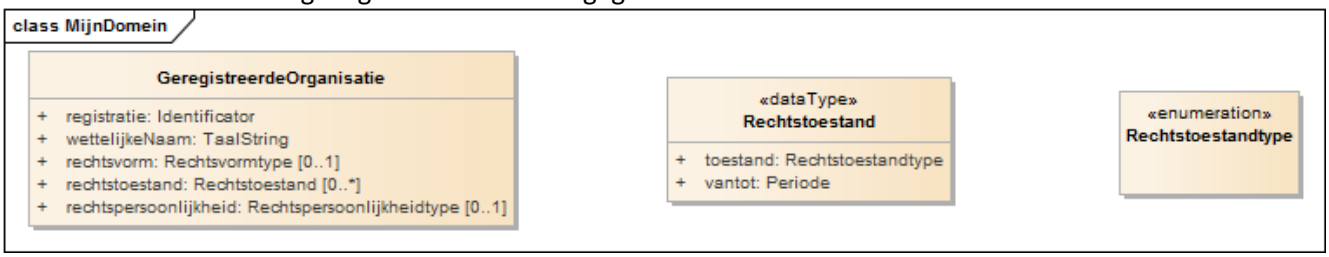
Bv bekijk volgend model:



Uitleg bij dit vb: Een GeregistreerdeOrganisatie heeft een bepaalde rechtstoestand (normale toestand, gerechtelijk akkoord, aanvraag faillissement etc). Dat is uiteraard de toestand op een bepaald moment, als we expliciet willen aangeven dat de organisatie verschillende toestanden heeft gekend gaan we de kardinaliteit moeten aanpassen (0..* ipv 0..1):

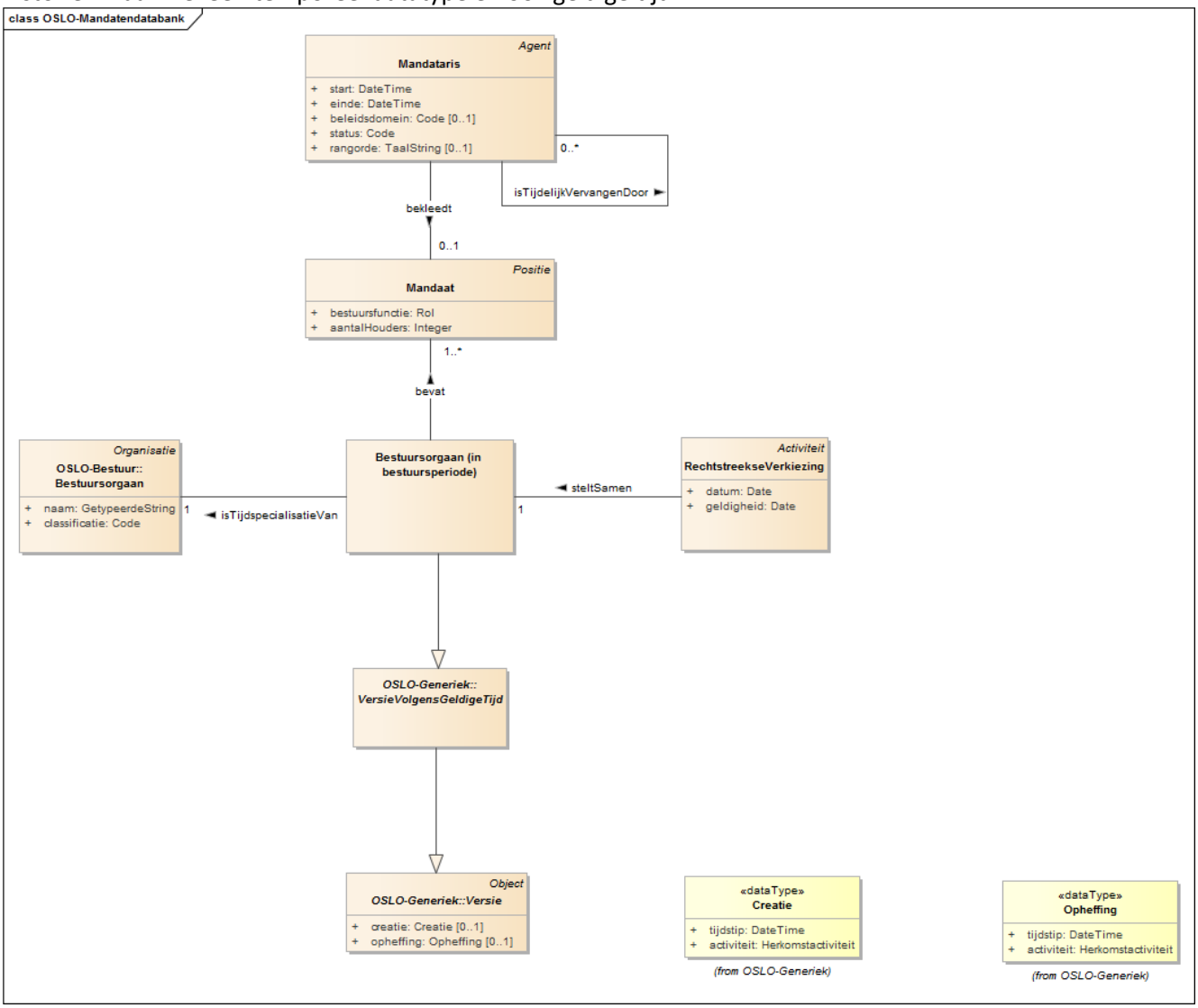
Wat tot semantische verwarring leidt: het lijkt of de GeregistreerdeOrganisatie op eenzelfde moment meerdere toestanden kan hebben.

De aangepaste kardinaliteit laat enkel toe om de voorgekomen toestanden weer te geven (nonsequenced historiek), willen we de opeenvolging van toestanden kennen (sequenced historiek) dan moet ook de periode waarbinnen de toestand geldig was worden meegegeven:



Resultaat is een klassendiagram dat semantische met implementatie-elementen mengt en daardoor moeilijk leesbaar wordt.

Opmerking: Dit betekent niet dat tijd niet mag voorkomen in een OSLO-model, in onderstaand vb staat geen historiek maar wel een temporeel datatype en ook geldige tijd:



Uitleg bij dit vb: Het volgende wordt hier gezegd over tijd:

- Een RechtstreekseVerkiezing vindt plaats op tijdstip t1.
- Daarbij wordt een Bestuursorgaan samengesteld voor een bepaalde periode (van creatie op t2 tot opheffing op t3).
- De opgegeven Mandaten gelden enkel voor het Bestuursorgaan in die bestuursperiode.
- Een Mandataris bekleedt een Mandaat van start t4 tot einde t5.

De historiek van deze gegevens, ttz de opeenvolgende bestuursperiodes en bijbehorende mandatarissen, werden echter niet gemodelleerd.

3.1.6 Versionering

Datastandaarden zijn onderhevig aan wijzigingen, we maken volgend onderscheid:

- Wijzigingen ah model.
- Wijzigingen aan de documentatie.

Enkel de eerste soort wijzigingen geeft aanleiding tot een nieuwe versie. Correcties van schrijffouten, oplossen van dode links, aanpassingen aan figuren ed hebben geen impact op applicaties die vd standaard gebruik maken.

Een standaard doorloopt verschillende levensfasen, waarbij doorgaans ook het model wordt aangepast. Ten minste elke levensfase brengt een nieuwe publicatie met zich mee en moet beschouwd worden als een nieuwe versie. Bv de overgang van In Ontwikkeling naar In Behandeling (= van Ontwerpdocument naar Kandidaat-Standaard) of van In Behandeling naar In Gebruik (van Kandidaat-Standaard naar Standaard).

Tijdens de fase In Ontwikkeling zijn meerdere herpublicaties mogelijk, di niet mogelijk wanneer de standaard In Behandeling of In Gebruik is.

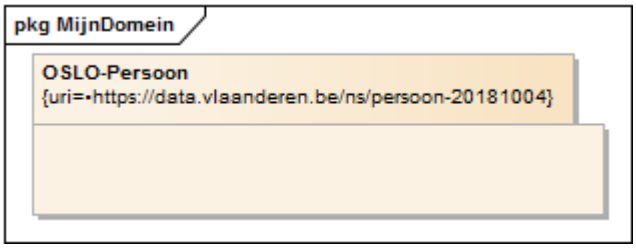
REGEL: Creëer een nieuwe versie bij elke publicatie, gebruik de publicatiedatum als versienummer.

Zie (Informatie Vlaanderen, 2018) voor meer info over de verschillende levensfases ve OSLO-standaard.

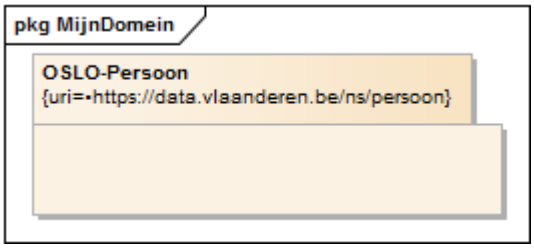
REGEL: Sla het versienummer op als metadata bij het model.

Zie §3.2.19 Metadata voor meer info.

Het versienummer komt voor in de documentatie (zie §3.1.11) en in de uri waarmee de standaard wordt aangeduid, vb



De laatste versie heeft een uri zonder publicatiedatum als alternatief, vb:



Met het versioneren vd modellen zijn impliciet ook de elementen die er deel van uitmaken geversioneerd, kortom:

REGEL: Ken geen expliciete versienummers toe aan de elementen ve model.

Is het model gepubliceerd als Standaard dan is het sowieso niet de bedoeling om nog iets aan individuele concepten te veranderen, enkel uitbreiding (bv van een klasse met een nieuw attribuut) is dan in principe nog mogelijk.

3.1.7 Afhankelijkheden

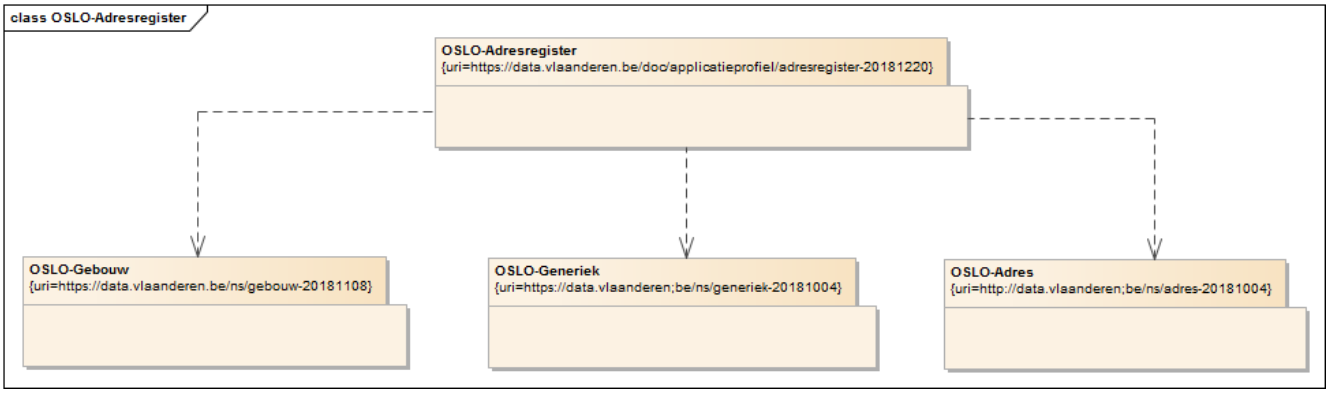
Door hergebruik van bestaande concepten zijn modellen onderling afhankelijk, ttz als een model evolueert heeft dit mogelijk impact op een ander model. Daarom:

REGEL: Documenteer de afhankelijkheid van andere modellen.

Vb door gebruikte VOC's op te sommen:

Vb door de namespace (het pakket) vd klasse te tonen ih UML-klassediagram:

Vb dmv een UML-package diagram:



Opmerking: We gaan er bij OSLO van uit dat de begrippen domein/model/pakket doorgaans samenvallen. Zie §3.2.23.

Opmerking 2: Merk op dat naar specifieke versies kan worden verwezen, bv in bovenstaand package diagram maakt het AP OSLO-Adresregister van 20/12/2018 gebruik van termen uit het VOC OSLO-Adres van 4/10/2018.

Opmerking 3: Mogelijk is er helemaal geen impact bij aanpassing ve bestaand model. Misschien wordt een concept gewijzigd dat niet door het andere model wordt gebruikt. Bovendien beperkt men zich eens een model gepubliceerd is als Standaard doorgaans tot uitbreidingen, zie §3.1.10.

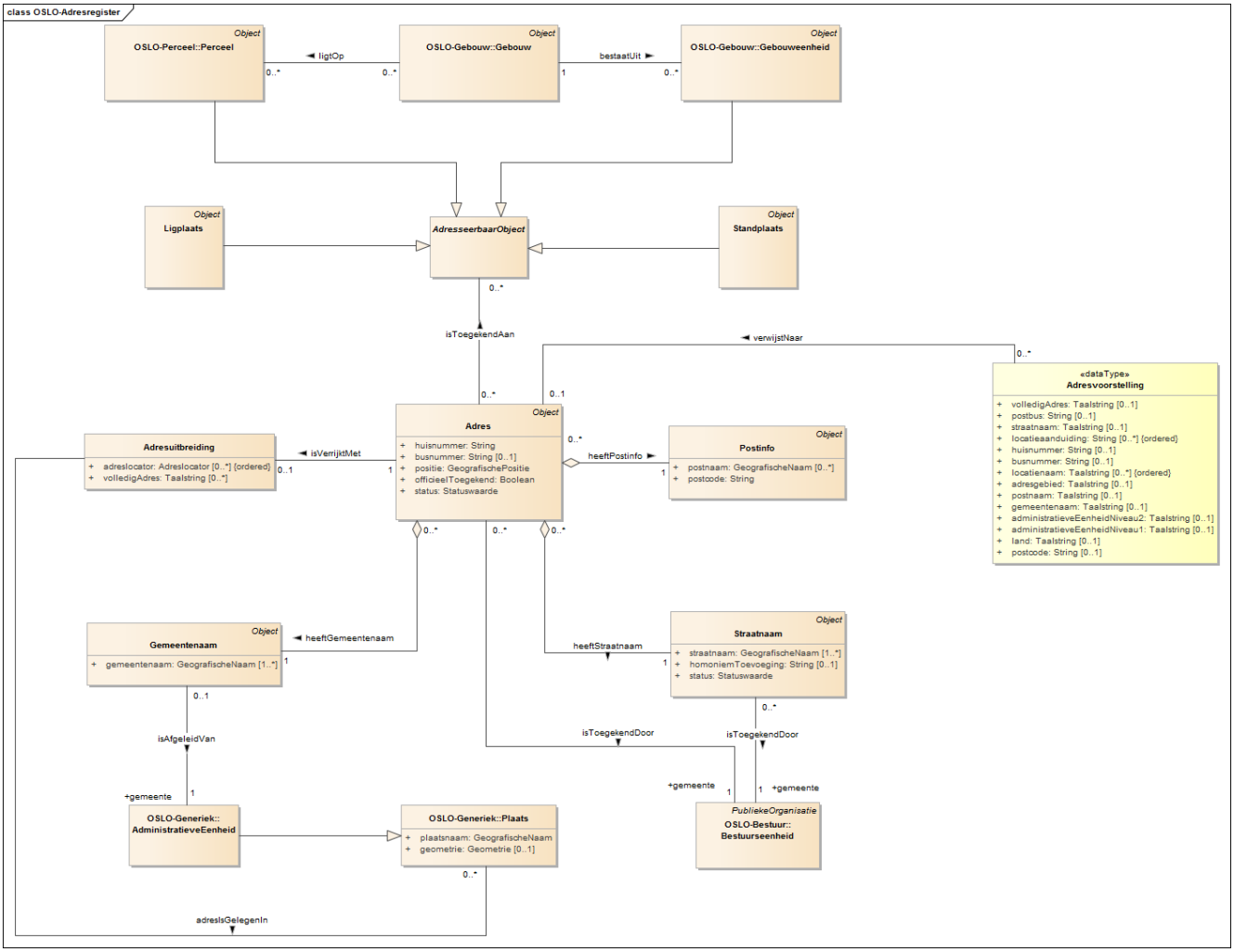
3.1.8 Schema

Een objectcataloog is ontoereikend om een model te beschrijven, het moet formeler. In praktijk betekent dit het gebruik ve modelleringstaal. Het resultaat noemen we een schema.

REGEL: Gebruik als modelleringstaal deze vh UML-klassendiagram.

We richten ons daarvoor op versie 2.5 van UML, zie (OMG, 2015).

Vb van een UML-klassendiagram:



Opmerking: OSLO ontwikkelde een profiel op UML, zie

Annex A: UML-profiel. Gebruik ervan is niet verplicht en kan evt beperkt blijven tot specifieke stereotypes. Handig is dat de stereotypes voorzien zijn van tags voor metadata, zie 3.2.19.

3.1.9 Standaardenregister

REGEL: Laat een nieuw model opnemen in het OSLO-Standaardenregister.

Dit kan zodra de standaard officieel in ontwikkeling is, meer info over de verschillende fasen in de levenscyclus ve OSLO-standaard in (Informatie Vlaanderen, 2018).

Het Standaardenregister biedt een overzicht van alle OSLO-standaarden met meta-informatie zoals:

- Ontwikkelingsstadium
- Categorie
- Verantwoordelijke organisatie
- Versie
- Einde publieke review periode (voor Kandidaat-Standaard)

En per standaard:

- Normatieve documentatie
- Niet-normatieve documentatie
- Verslagen van werkgroepvergaderingen
- Presentaties en ander materiaal
- Detailinformatie

Het overzicht geeft een beeld van welke standaarden beschikbaar zijn, hoe recent ze zijn (versie adhv publicatiedatum) en of ze In Ontwerp, In Behandeling of al In Gebruik zijn (resp Ontwerpdocument, Kandidaat-Standaard of Standaard).

Daarna wordt verwezen naar de specificatie vd standaard (de normatieve documentatie), aangevuld met niet-normatieve documentatie (bv hoe de standaard implementeren, objectcataloog voor specifiek gebruik etc).

Het tot stand komen vd standaard kan adhv verslagen vd werkgroepvergaderingen gereconstrueerd worden.

Indien beschikbaar worden presentaties, webcasts etc toegevoegd die de normatieve documentatie inzichtelijker kunnen maken.

Detailinformatie geeft ten slotte een zicht op het officieel traject (aanmeldingsdatum, erkenningsdatum etc.).

Het OSLO-Standaardenregister is te vinden op volgend adres: https://data.vlaanderen.be/standaarden.

3.1.10 Uitbreiding

Onder uitbreiding verstaan we het toevoegen van concepten aan een VOC. Er zijn volgende mogelijkheden:

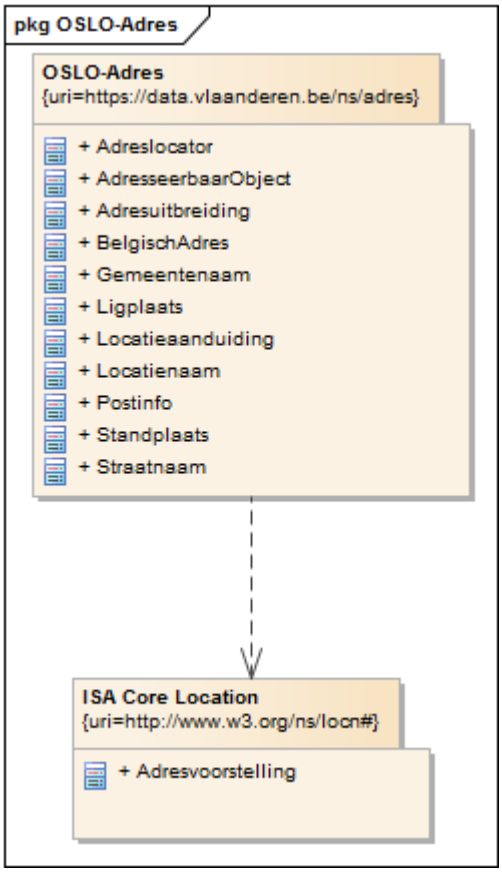
1. Uitbreiding vh eigen VOC
2. Uitbreiding ve bestaand VOC

Het eerste geval spreekt voor zich: men kan altijd elementen (klassen, attributen, associaties…) toevoegen aan een model dat men zelf beheert. Wel moeten daarbij de regels vh ontwikkelproces worden gevolgd, ttz door de uitbreiding is een officiële herpublicatie en versieverhoging noodzakelijk en zal het VOC van In Gebruik naar In Revisie gaan, zie (Informatie Vlaanderen, 2018).

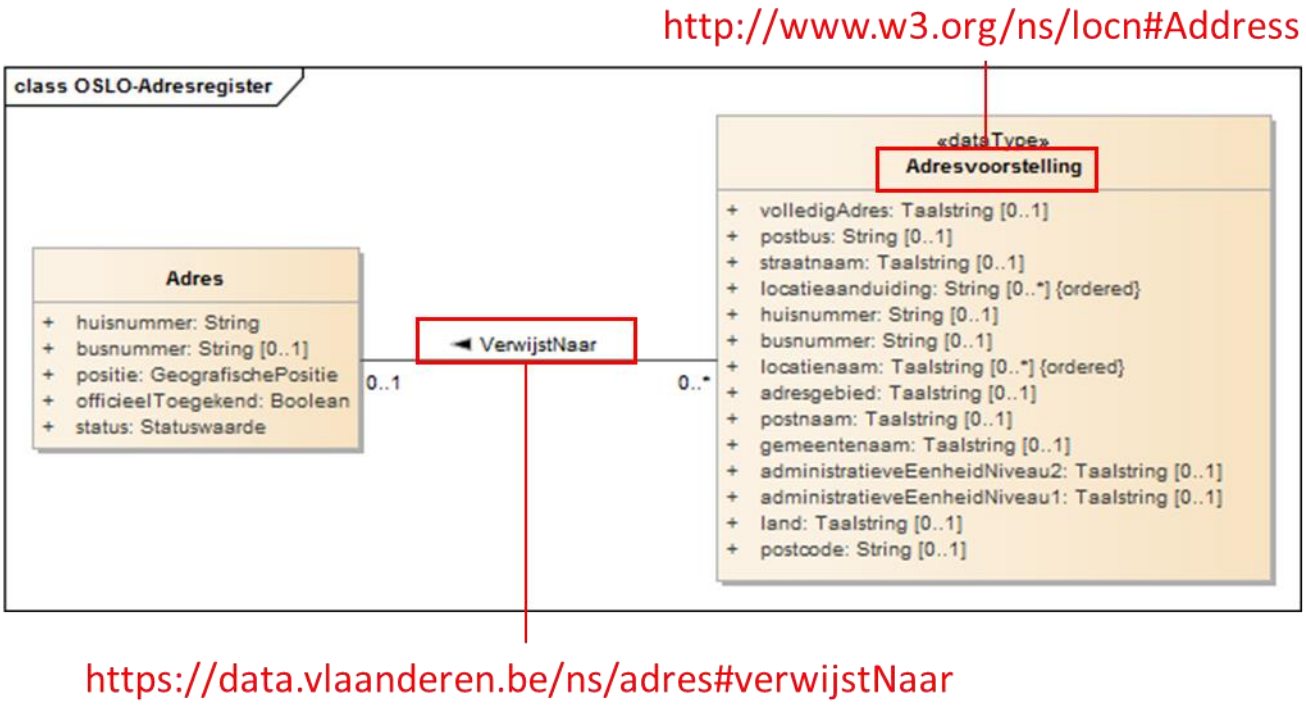
Het tweede geval wordt ingegeven door de Open World Assumption (OWA) vh semantisch web, zie §2.3. Die stelt dat iedereen elementen aan bestaande VOC's kan toevoegen, wat in praktijk echter neerkomt op het creëren ve nieuw VOC.

REGEL: Creëer een nieuw VOC om een bestaand VOC uit te breiden.

Bv we maakten een VOC OSLO-Adres om het VOC ISA Core Location uit te breiden met klassen zoals Straatnaam, BelgischAdres etc:



Het is ook mogelijk om attributen en associaties toe te voegen aan bestaande klassen, bv voegden we aan het datatype Adresvoorstelling uit ISA Core Location een associatie VerwijstNaar toe:



Op dezelfde manier kan ook extra metadata toegevoegd worden aan elementen in een bestaand VOC, bv een vertaling naar het Nederlands vh oorspronkelijk Engelstalig label. Bv attributen vd klasse Address uit het VOC ISA Core Location krijgen in het VOC OSLO-Adres een Nederlandstalig label, bv “volledig adres” voor “full address”.

Metadata wordt in UML toegevoegd dmv “tagged values”, meer daarover en over welke metadata kan worden toegevoegd in §3.2.19.

3.1.11 Documentatie

Onder “documentatie” verstaan we de normatieve documentatie (ook wel specificatie genoemd) vd datastandaard. In het Standaardenregister (zie §3.1.9) wordt ten minste de normatieve documentatie opgenomen, maar ook niet-normatieve documentatie zoals implementatievoorbeelden edm.

REGEL: Neem de voorgeschreven info op in de normatieve documentatie.

Afhankelijk of het over een VOC of een AP gaat bevat de specificatie ten minste volgende onderdelen:

VOC	AP
titel	titel
versie	versie
laatste wijziging	laatste wijziging
uri laatste/huidige/vorige versie	uri laatste/huidige/vorige versie
auteurs/editors/medewerkers	auteurs/editors/medewerkers
samenvatting	samenvatting
status	status
licentie	licentie
conformiteit	conformiteit
	klassendiagram
objectcataloog	objectcataloog

De versie wordt aangeduid door de publicatiedatum vh model, zoals vermeld in §3.1.6. De laatste wijziging is eveneens een datum, ttz de datum waarop de laatste (editoriale) wijziging plaatsvond.

De uri vd laatste versie is deze zonder publicatiedatum, die van de huidige en vorige versies met publicatiedatum (meer over deze uri's in §3.1.6).

Status heeft betrekking op de status vh model (Ontwerpdocument, Kandidaat-Standaard, Standaard etc) afhankelijk vd levensfase waarin het zich bevindt (In Ontwikkeling, In Behandeling, In Gebruik etc). Meer info daarover in (Informatie Vlaanderen, 2018).

Een model moet formeel worden beschreven door een schema, in praktijk een UML-klassendiagram zoals vermeld in §3.1.1 en 3.1.8. Voor VOC's vervalt deze vereiste momenteel, we volstaan met de evenzeer formele (maar niet-grafische) beschrijving in RDF (in principe bekomen door het UML-klassendiagram om te zetten naar RDFS/OWL).

De objectcataloog bevat ten minste volgende info per element:

VOC	AP
label	label
uri	uri
datatype (tenzij bestaand element)	datatype
definitie (tenzij bestaand element)	definitie
gebruik (tenzij bestaand element)	gebruik
	kardinaliteit
	enumeratie

Label en uri moeten zowel in de documentatie ve VOC als van een AP steeds voor elk element worden opgegeven.

Datatype, definitie en gebruik moeten bij een VOC voor externe elementen niet worden meegegeven in het model (zie §3.2.19) en ontbreken in dat geval ook in de documentatie. Dat geldt niet voor AP's waar de definitie van externe elementen wordt verfijnd ihkv de toepassing die het AP beschrijft.

Kardinaliteiten en enumeraties komen in VOC's niet voor (zie inhoud ve model in §3.1.1) en dus ook niet in de documentatie.

Kijk in het OSLO-Standaardenregister voor voorbeelden van normatieve documentatie, meer info in §3.1.9.

In principe zou eenzelfde model eenzelfde klassendiagram moeten opleveren, echter worden soms wijzigingen aangebracht louter om de leesbaarheid vd specificatie te verhogen:

REGEL: Pas het klassendiagram eventueel aan omwille vd leesbaarheid.

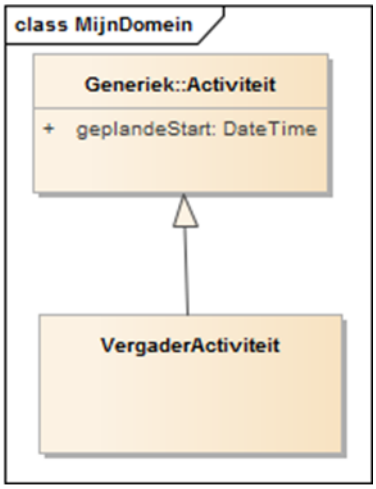
Opmerking: Deze aanpassingen mogen geen impact hebben op het model zelf.

Mogelijke aanpassingen:

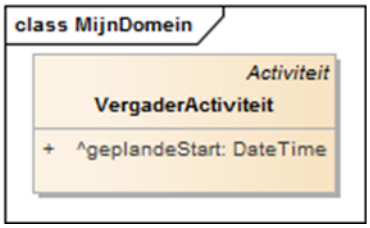
- Weglaten van superklassen
- Weergeven van overgeërfde elementen
- …

Vb:

VOC



AP



Uitleg bij dit vb: De superklasse Activiteit is weggelaten in het klassendiagram vh AP omwille van de leesbaarheid. VergaderActiviteit is een Activiteit en erft dus het attribuut geplandeStart over maar eens de superklasse is weggelaten is dit niet meer zichtbaar. Als oplossing biedt UML de mogelijkheid om het attribuut

eveneens toe te voegen aan de subklasse, echter voorafgegaan door het caret-symbool om aan te geven dat het een overgeërfd kenmerk is, zie §3.2.5 voor meer info. Het attribuut behoudt daarbij zijn uri, in het vb https://data.vlaanderen.be/ns/generiek#geplandeStart.

3.2 MODEL – EN SCHEMAREGELS

De regels hebben betrekking op:

- Metamodel
- Klassen en attributen
- Generalisatie/specialisatie
- Generalisatieset
- Overerving
- Abstractie
- Soorten associaties
- Soorten datatypes
- Klasse of datatype?
- Associatierollen, navigatie
- Attribuut of associatie?
- Associatieklassen
- Specialisatie van associaties
- Multipliciteit
- Constraints
- Enumeraties
- Subklasse of enumeratie?
- Metadata
- Naamgeving
- Definities
- Zichtbaarheid
- Pakketten

Opmerking: Meestal hebben de regels betrekking zowel op het modelleren zelf (herkennen vd concepten) als op de schematische voorstelling ervan. Om die reden maken we geen onderscheid tussen beide.

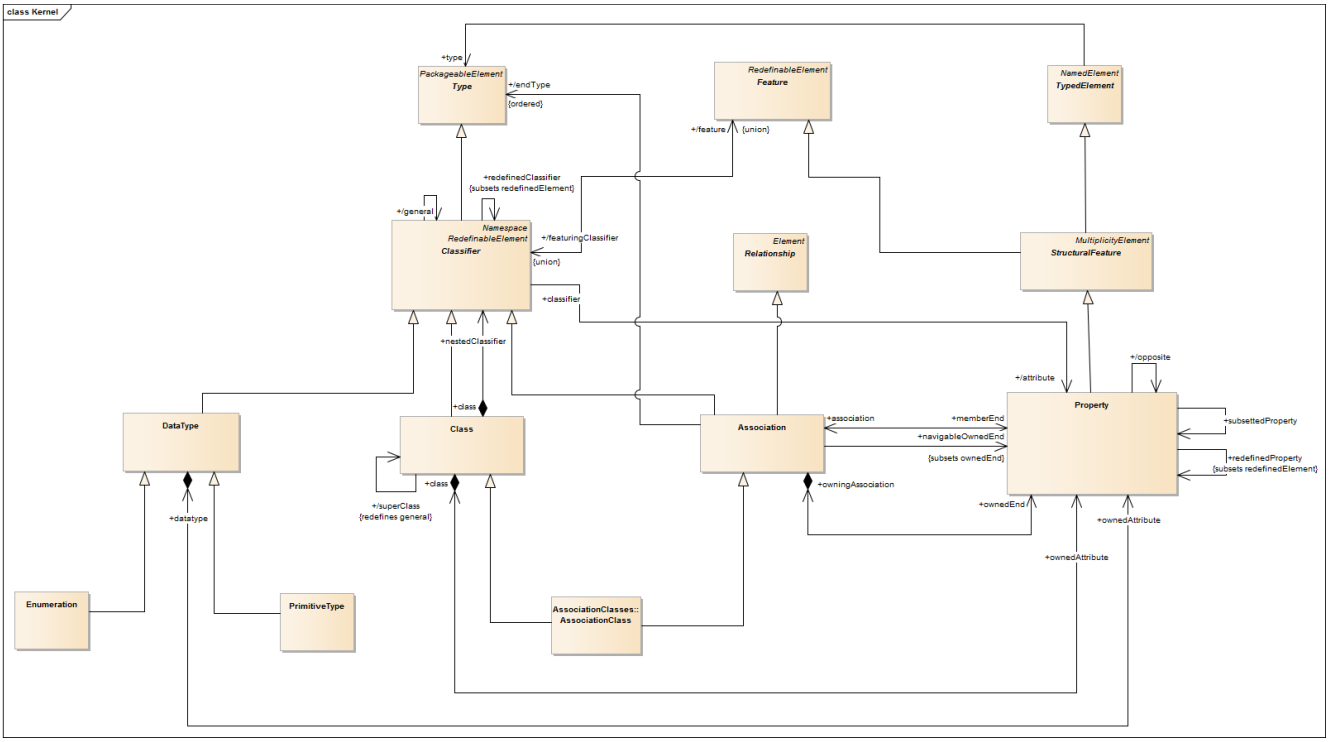
3.2.1 Metamodel

REGEL: Modelleer met de elementen uit het UML-metamodel voor klassen.

Onder metamodel verstaan we een model vh model, een model dat de kenmerken beschrijft vh model zelf. Een metamodel beschrijft dus de modelleringstaal, hier UML.

UML is erg ruim, we beperken ons hier tot UML-Klassen, zoals deze doorgaans worden gebruikt in een UML-klassendiagram.

Typisch wordt een metamodel zelf voorgesteld dmv een UML-klassendiagram:

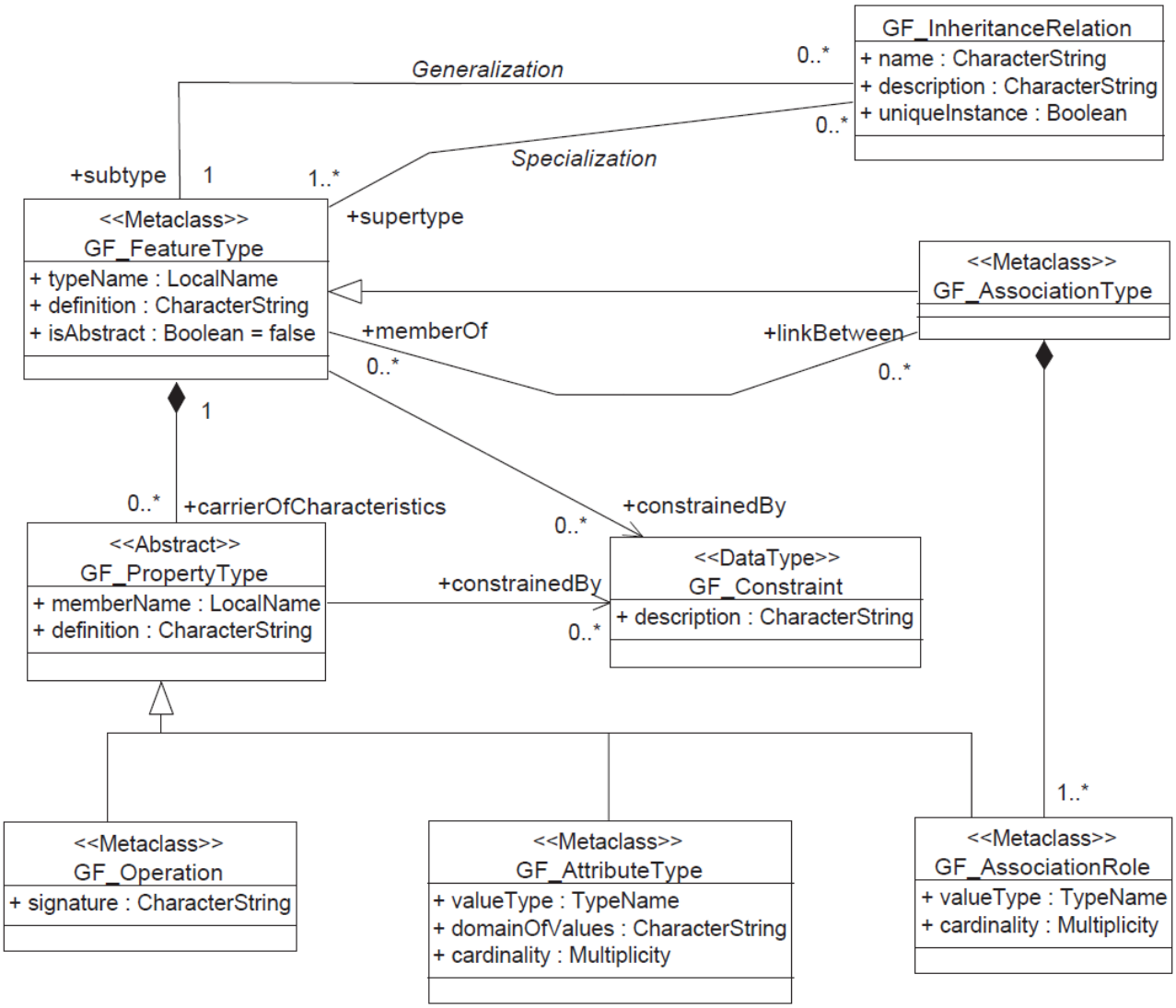


Opgebouwd op basis van het xmi van het UML metamodel (SPARX)

In woorden:

- Een model kan bestaan uit Klassen, Datatypes en Associaties
- Klassen en Datatypes kunnen Attributen hebben
- Associaties ook maar we noemen ze dan Associatie-uiteinden
- Een Enumeratie is een Datatype
- Attributen hebben een type, verder specialiseerbaar als Klasse, Datatype, Enumeratie…
- Attributen van Klasse of Datatype zijn “owned” en vervallen samen met de klasse of het datatype
- Als een Klasse vervalt, vervallen ook zijn subklassen
- Etc

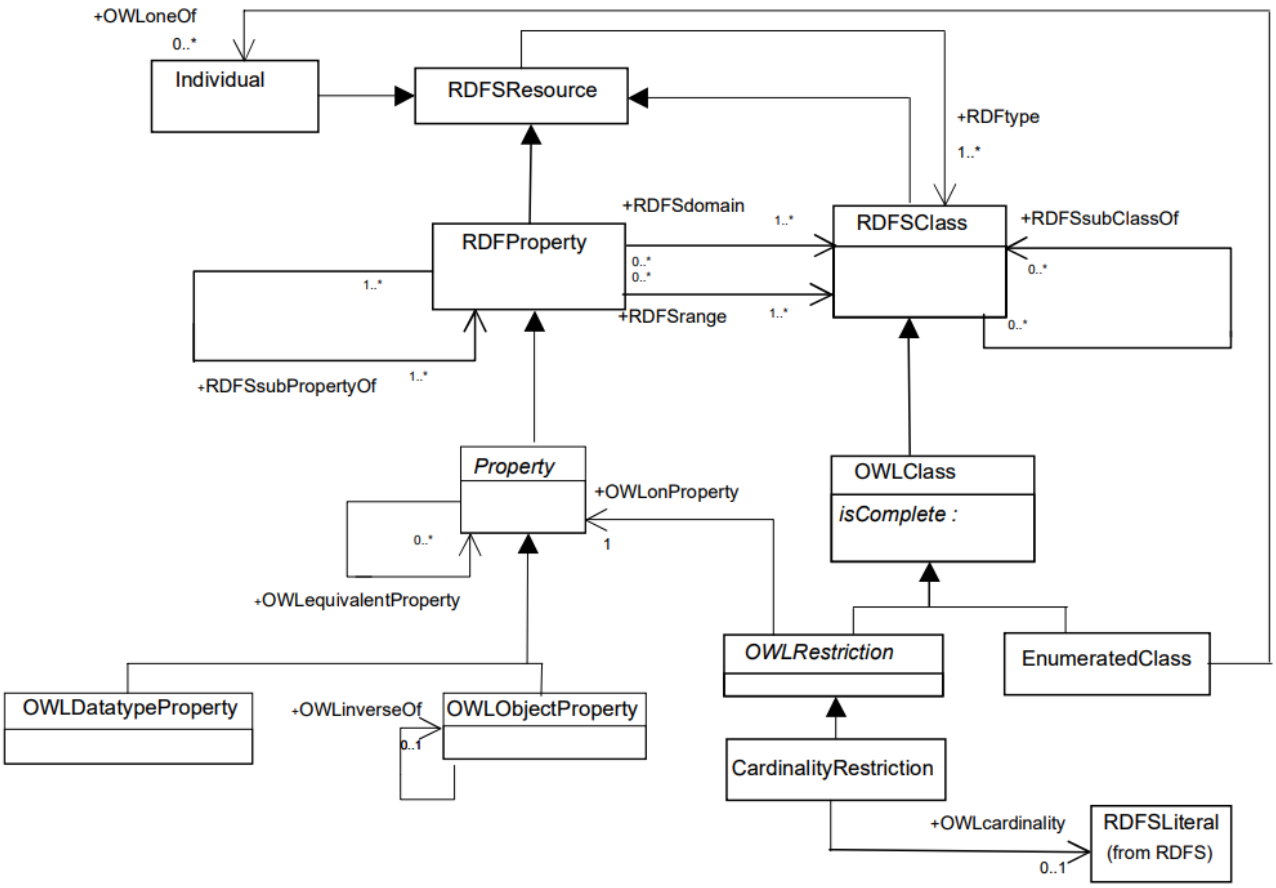
Vergelijk met het zgn General Feature Model (GFM) vd ISO TC211:



Bron: (ISO 19109, 2015), p12.

Opmerking: INSPIRE heeft een profiel op dit GFM, het zgn Generic Conceptual Model (GCM).

Vergelijk ook met de RDFS/OWL-metamodellen voor het modelleren van vocabularia/ontologieën. Ook daarvan bestaat een UML-profiel, het Ontology Definition Metamodel (ODM) van OMG. Dat ziet er als volgt uit:

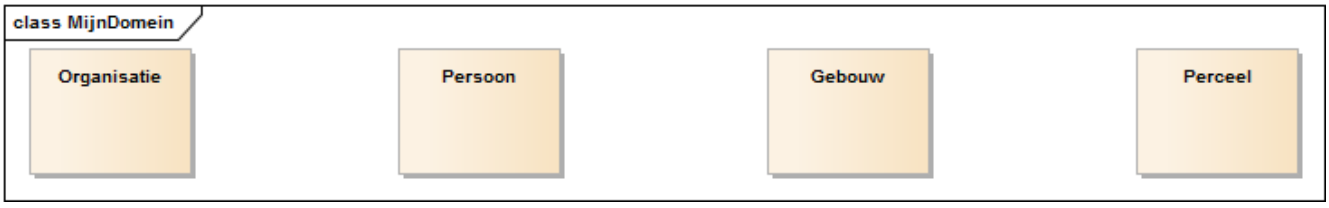


Bron: (Colomb, 2006)

Opmerking: Bovenstaande figuur is een sterke vereenvoudiging, zie (OMG, 2014) voor het volledige ODM-metamodel.

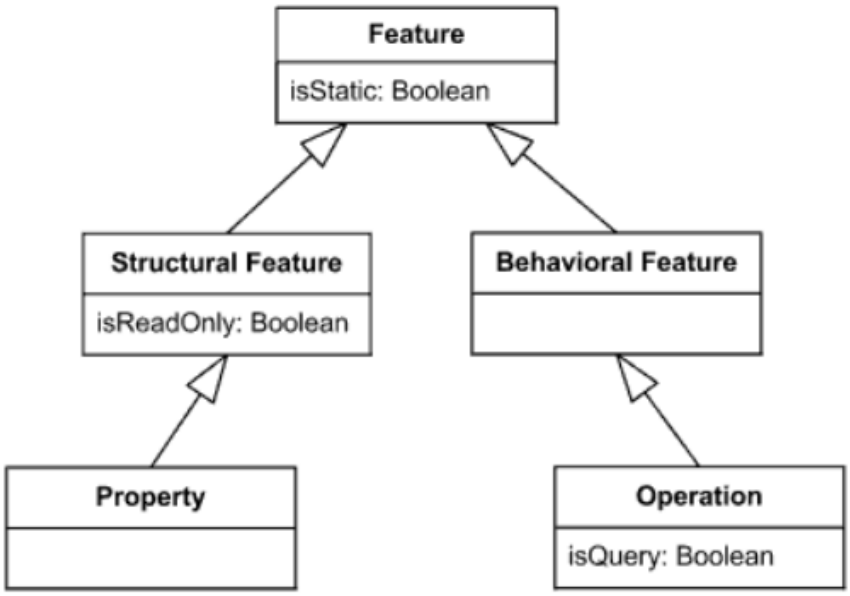
3.2.2 Klassen en attributen

Een klasse is een zgn classifier en beschrijft een verzameling objecten met dezelfde kenmerken. Enkele vb:



Ter info: andere classifiers zijn: datatypes en associaties, waarbij een datatype een verzameling waarden met dezelfde kenmerken beschrijft en een associatie een verzameling relaties met dezelfde kenmerken (zie §3.2.1).

“Kenmerken” (features in het Engels) zijn erg ruim:

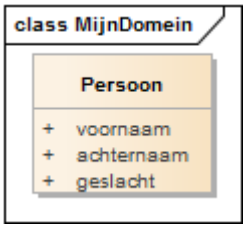


Bron: (Fakhroutdinov)

Maw: het gaat zowel over structurele kenmerken zoals attributen als over behavioral kenmerken zoals operaties.

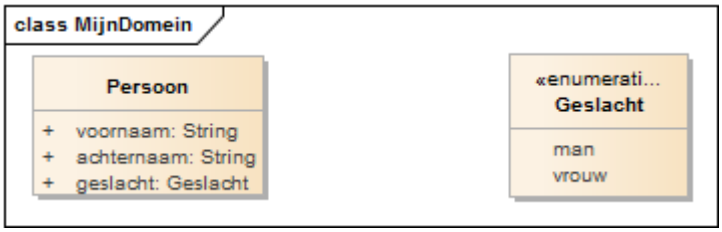
Voor de volledigheid: naast kenmerken spelen ook betekenis (semantiek) en beperkingen (constraints) een rol bij het klasseren van objecten, zie (Fakhroutdinov).

Een attribuut is een bepaald kenmerk ve klasse of datatype, vb:



Een attribuut heeft een type (zie §3.2.1), dat op zijn beurt een classifier is en doorgaans een datatype (als het een klasse is kan ook een associatie worden overwogen, zie §3.2.12).

Vb met datatypes:



Geslacht in bovenstaand vb is een enumeratie (een enumeratie is een gespecialiseerd datatype, zie §3.2.17).

REGEL: Geef van een attribuut steeds het datatype op.

Dit is van belang bij de uiteindelijke implementatie vh conceptueel model of applicatiemodel.

3.2.3 Generalisatie/specialisatie

Generalisatie betekent het opvoeren van een concept dat een veralgemening is van een ander concept. Het algemeen concept (de superklasse) heeft kenmerken die gelden voor elk oorspronkelijk concept (de subklasse).

REGEL: Let er op dat de attributen, associaties etc van een superklasse van toepassing zijn op al zijn subklassen.

Bv als een superklasse Organisatie een naam heeft, dan heeft impliciet ook zijn subklasse PubliekeOrganisatie een naam, dit door het zgn overerven.

Meer info over overerving in §3.2.5.

Tussen superklasse en subklasse bestaat een “is een” associatie, in voornoemd vb geldt dus: een PubliekeOrganisatie is een Organisatie. Deze relatie moet niet worden benoemd.

REGEL: Kijk na of er tussen subklasse en superklasse een “is een” relatie bestaat.

Opmerking: Dit betekent concreet dat een superklasse mag worden vervangen door zijn subklasse in uitwisseling.

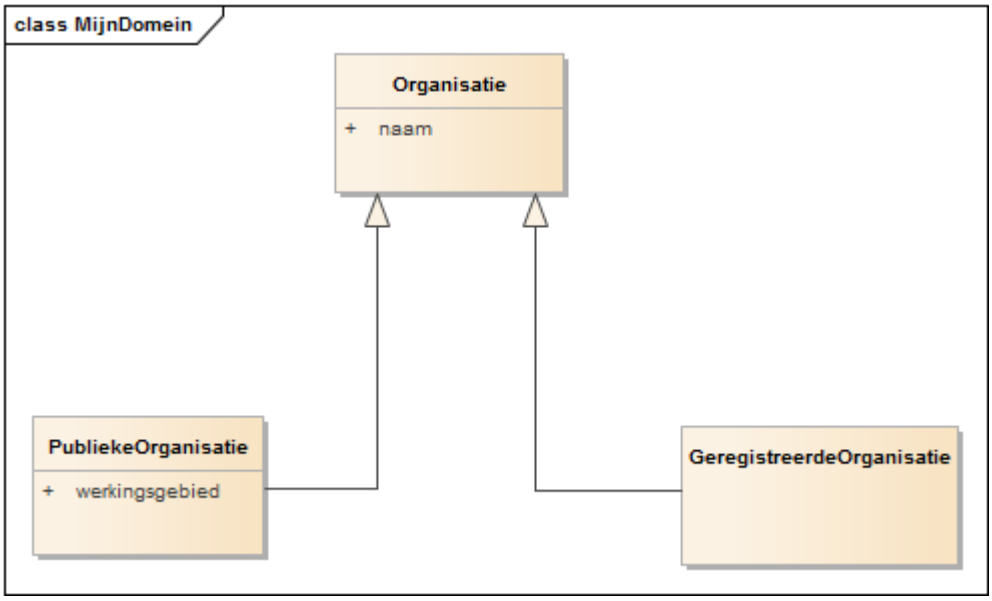
Omgekeerd is specialisatie het opvoeren van een concept dat specifieker is dan een bestaand concept.

Zie §3.2.18 voor de overweging om in de plaats van subklassen een enumeratie te gebruiken.

Zie §3.2.14 voor het specialiseren van attributen en associaties.

3.2.4 Generalisatieset

Specialisatie volgens verschillende invalshoeken is mogelijk. Zo kan een Organisatie verder gespecifieerd worden als een PubliekeOrganisatie maar ook als een GeregistreerdeOrganisatie:



Uitleg bij dit vb: Beide subklassen overlappen, ze sluiten elkaar niet uit: maw een PubliekeOrganisatie kan tegelijk ook een GeregistreerdeOrganisatie zijn.

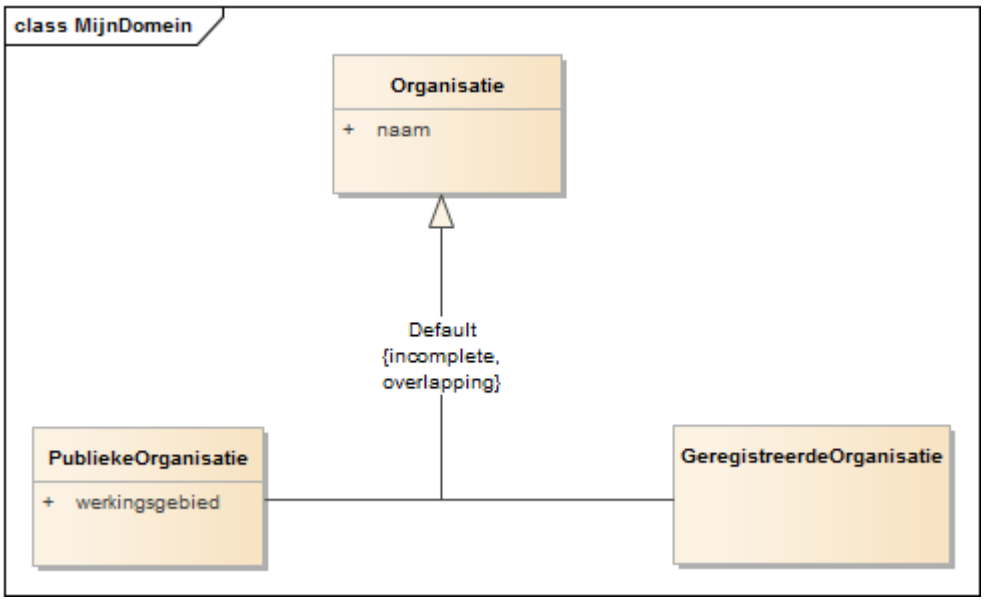
Classificaties volgens een bepaalde invalshoek hoeven ook niet uitputtend te zijn, bv als GeregistreerdeOrganisaties bestaan, bestaan ook NietGeregistreerdeOrganisaties maar het is niet nodig om deze subklasse op te voeren.

Default in UML is onvolledig & overlappend, zie (OMG, 2015) p119.

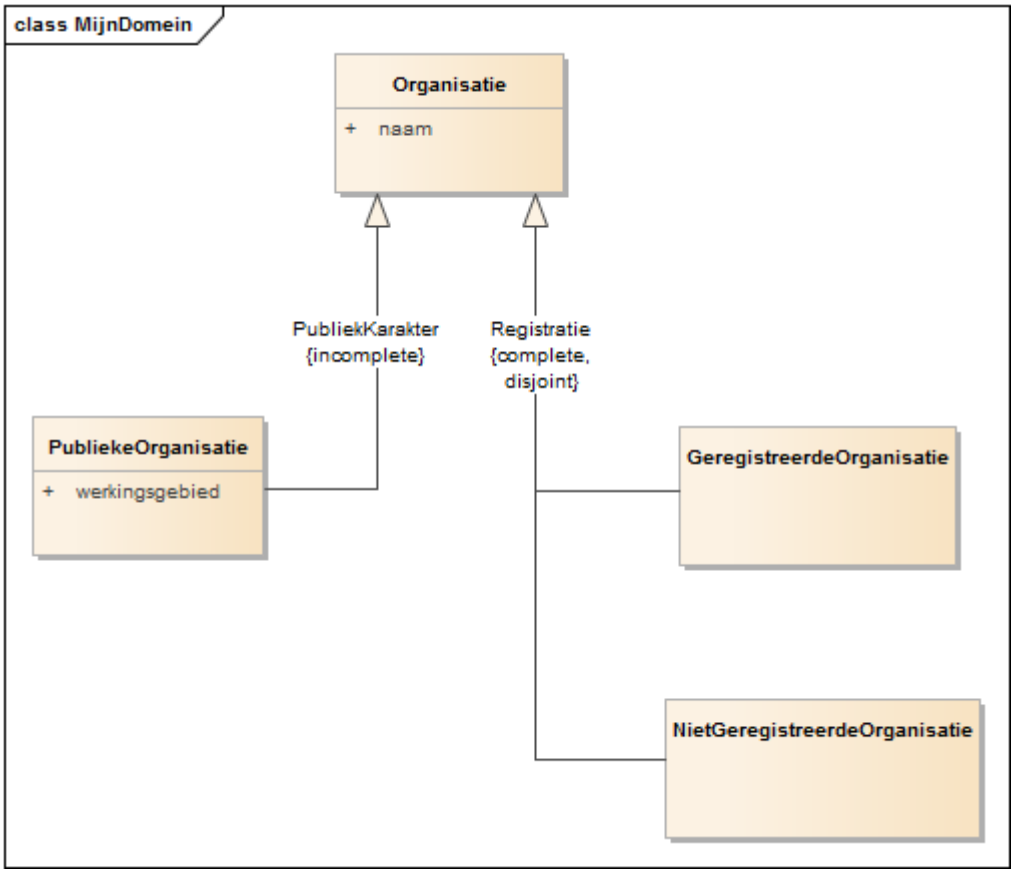
Opmerking: Het is pas sinds UML 2.5 dat dit de default is, zie (Fakhroutdinov).

Opmerking 2: UML kent een concept “generalisatieset” dat toelaat om de invalshoeken volgens dewelke subklassen werden gecreëerd expliciet van elkaar te onderscheiden. Per generalisatieset kan men aangeven of de overeenkomstige classificatie al dan niet volledig is (complete/incomplete) en of de subklassen elkaar al dan niet uitsluiten (disjoint/overlapping).

In voorgaand vb is de default generalisatieset van toepassing:



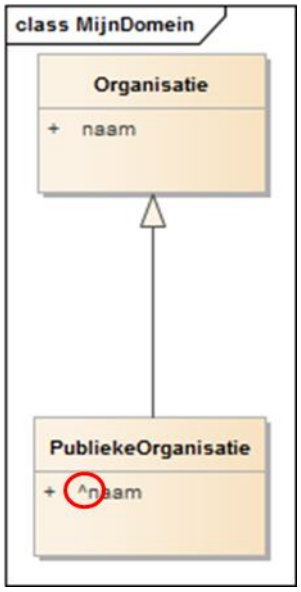
Maar we zouden deze ook kunnen opsplitsen in twee generalisatiesets bv als volgt:



3.2.5 Overerving

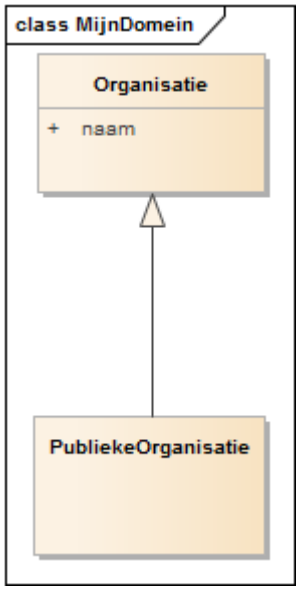
Bij specialisatie (zie §3.2.3) worden niet enkel attributen maar ook andere kenmerken vd superklasse overgeërfd waaronder associaties & constraints.

Vb overerving attribuut:

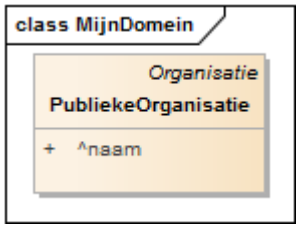


Waarbij het dakje (^ of caret in het Engels) voor de naam aangeeft dat het een overgeërfd kenmerk betreft.

Opmerking: Het is niet verplicht om overgeërfde kenmerken expliciet weer te geven, maw onderstaande figuur is equivalent:

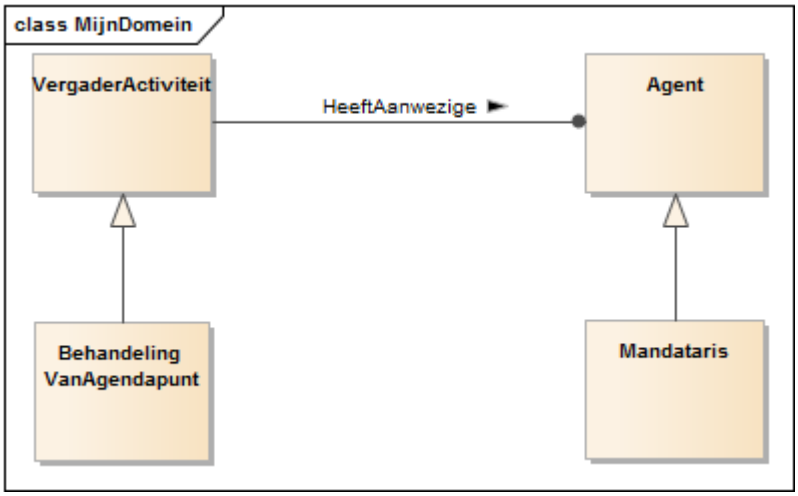


Het expliciet weergeven van overgeërfde attributen is vooral handig als bv om reden van leesbaarheid de superklasse uit het diagram wordt weggelaten:

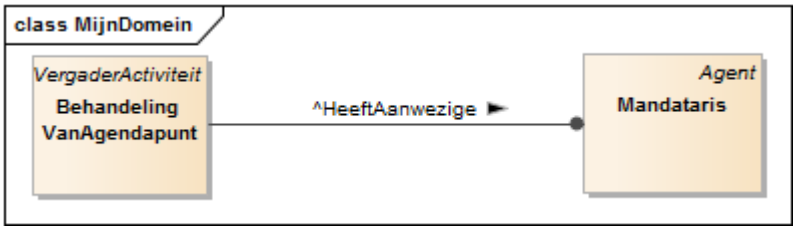


REGEL: Toon overgeërfde kenmerken als de superklasse niet zichtbaar is in het diagram.

Vb overerving associatie:



Of:

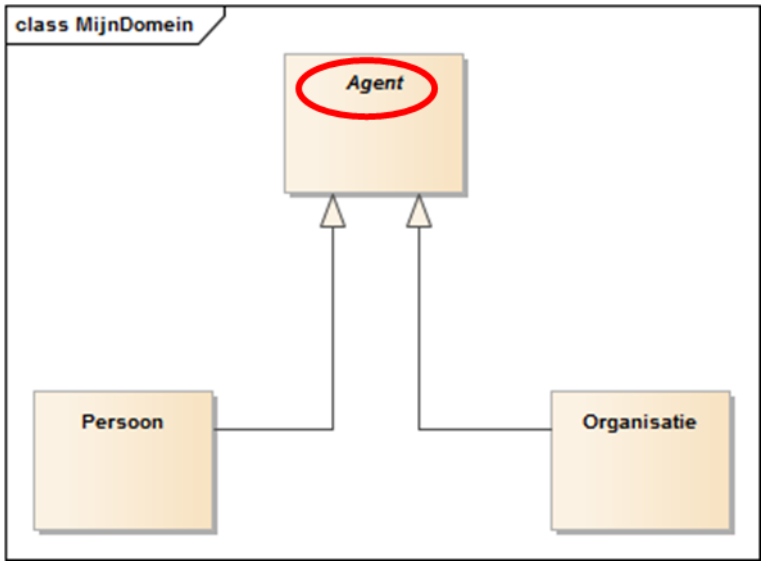


Opmerking: uiteraard blijft de uri vh concept hetzelfde bij overerving, in bovenstaand vb https://data.vlaanderen.be/ns/besluit#HeeftAanwezige.

3.2.6 Abstractie

Abstractie geeft aan dat een concept niet materieel is.

Abstracte klassen zijn herkenbaar aan het feit dat hun naam schuingedrukt wordt weergegeven op het klassendiagram, vb:



Abstracte klassen zijn bijproducten vh abstractieproces dat modelleren is, ze ontstaan door concepten te groeperen tot meer algemene concepten (zie ook §3.2.2) die in de materiële werkelijkheid niet bestaan.

REGEL: Maak een superklasse abstract om aan te geven dat in praktijk zijn subklassen moeten worden gebruikt.

In voorgaand vb betekent dit dat bij data-uitwisseling gespecifieerd moet worden of de instantie vd klasse betrekking heeft op een Persoon of op een Organisatie. Maw: direct naar een Agent verwijzen is niet mogelijk maar indirect is een Persoon of Organisatie wel steeds een Agent.

3.2.7 Associaties

Een associatie is een semantische relatie tussen klassen, zie (OMG, 2015) p197. Het betreft een relatie die een verband aangeeft tussen concepten op basis van hun betekenis, bv:

Uitleg bij dit vb: Een PubliekeDienstverlening heeft doorgaans een PubliekeOrganisatie die er verantwoordelijk voor is, deze relatie is duidelijk inhoudelijk en niet technisch.

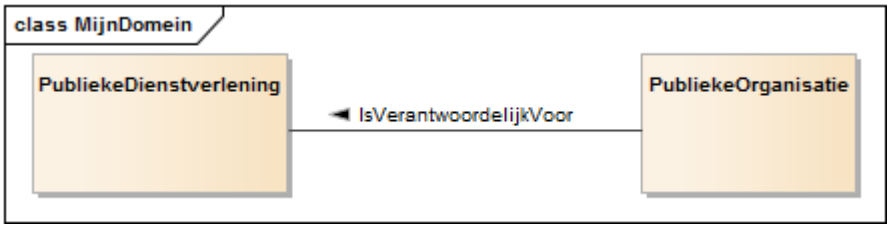
Default wordt een associatie weergegeven dmv een volle lijn tussen de verbonden klassen met de naam van de relatie in het midden.

UML voorziet de mogelijkheid om de leesrichting vd associatienaam mee te geven dmv een volle pijl:

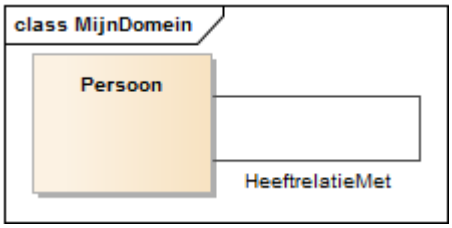
REGEL: Geef indien nodig de leesrichting vd associatienaam weer.

In bovenstaand vb is dit noodzakelijk, een PubliekeDienstverlening heeft een verantwoordelijke PubliekeOrganisatie maar niet omgekeerd.

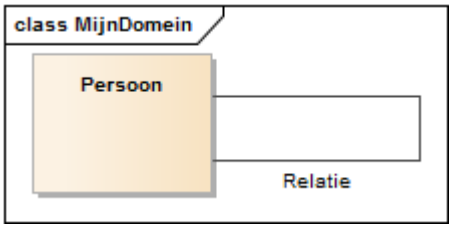
Opmerking: Dit betekent niet dat de inverse relatie niet bestaat, alleen zou deze dan een andere naam moeten krijgen, vb in bovenstaand geval:



Voorbeeld ve geval waarbij opgave vd leesrichting niet noodzakelijk is:



Opmerking: In bovenstaande voorbeelden beginnen de associatienamen met een persoonsvorm. Dat heeft het voordeel dat het geheel dan op een zin lijkt, bv “Persoon HeeftRelatieMet Persoon” wat de leesbaarheid vh diagram bevordert en analoog is met de impliciete “IsEen” relatie bij specialisatie (zie §3.2.2). Dit is echter niet verplicht, bv ook het volgende is mogelijk:

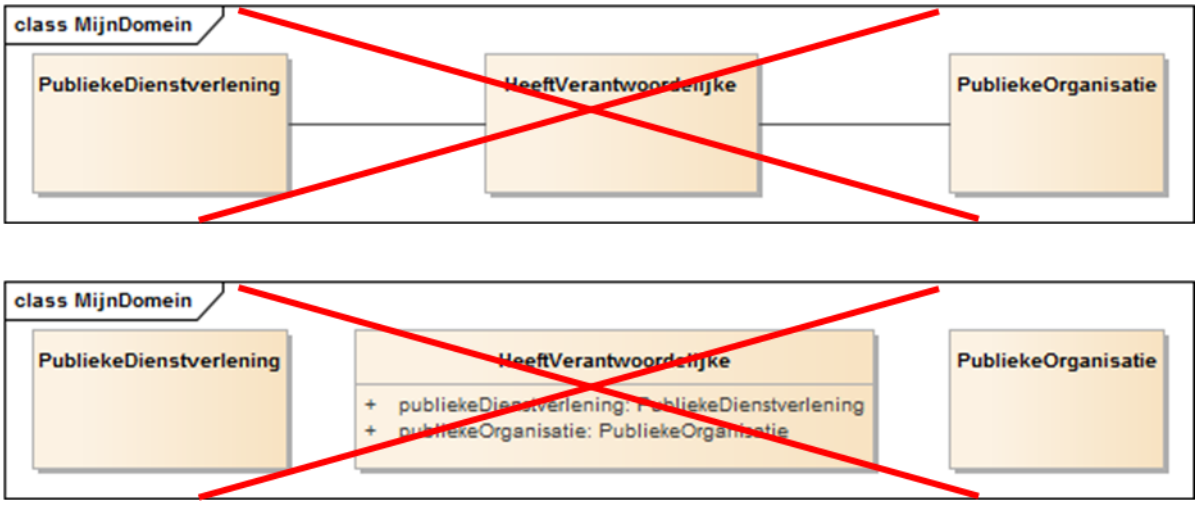


Op de naam vd associatie zijn de naamgevingsconventies van toepassing (zie §3.2.20).

Opmerking: Associaties zijn classifiers (zie metamodel in §3.2.1), ttz een soort klassen waarbij de uiteinden vd associatie (die naar de klassen verwijzen die de associatie met elkaar verbindt) eigenlijk attributen zijn. Bij implementatie bestaat maw de kans dat de associatie werkelijk een klasse wordt (tenzij de verbonden klassen eigenaar zijn vd associatie-uiteinden, zie §3.2.11). Dat betekent in geen geval dat daarop vooruitlopend de associatie als klasse moet worden gemodelleerd:

REGEL: Materialiseer de associatie niet als klasse.

Vb:



3.2.8 Aggregaat/composiet

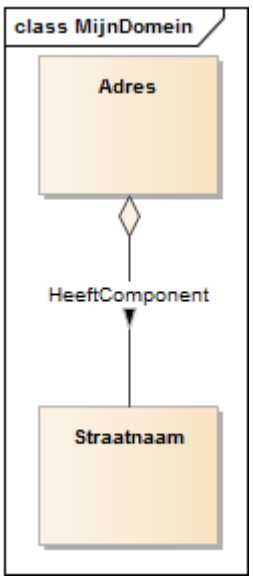
De concepten die een associatie met elkaar verbindt zijn in principe evenwaardig. Is dat niet het geval en is er sprake van een geheel-onderdeel relatie dan zijn er twee mogelijkheden

- Aggregaat
- Composiet

In beide gevallen stelt één klasse het geheel voor en één of meer andere klassen het onderdeel of de onderdelen waaruit het geheel bestaat.

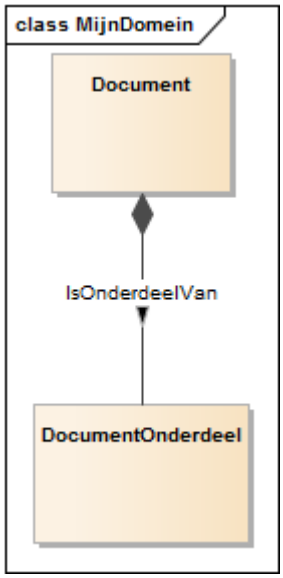
Is de relatie tussen geheel en onderdeel zwak, dan spreken we van een aggregaat. De relatie is zwak als het onderdeel verder kan blijven bestaan als het geheel wegvalt, sterk als de onderdelen verdwijnen als het geheel verdwijnt.

Vb van een aggregaat:



Uitleg bij dit vb: Een straatnaam is een component ve adres, maw samen met de andere componenten is het een onderdeel ve adres. Maar dat betekent niet dat de straatnaam verdwijnt als het adres verdwijnt.

Vb van een composiet:



Uitleg bij dit vb: Een document bestaat uit een aantal onderdelen: hoofdstukken, paragrafen etc. In dit geval is de geheel-onderdeel relatie sterker: als het document wegvalt bestaan ook de onderdelen waaruit het bestaat niet meer.

REGEL: Gebruik een aggregaat-associatie in het geval van een zwakke geheel-onderdeel relatie en een composiet-associatie in het geval van een sterke geheel-onderdeel relatie.

3.2.9 Datatypes

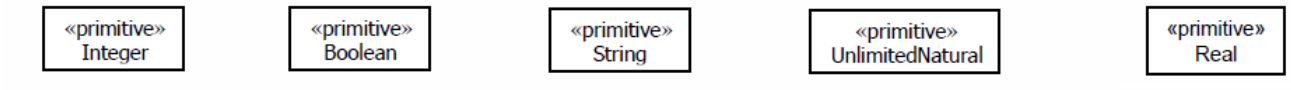
Een datatype is een verzameling waarden met dezelfde kenmerken (vgl met de definities van de andere classifiers in §3.2.2).

We maken volgend onderscheid:

- Primitief datatype
- Enumeratie
- Gestructureerd datatype

Primitieve datatypes hebben geen attributen, ze vertegenwoordigen direct de waarden die ze voorstellen.

Standaard voorziet UML volgende primitieve datatypes:



Bron: (OMG, 2015) p675.

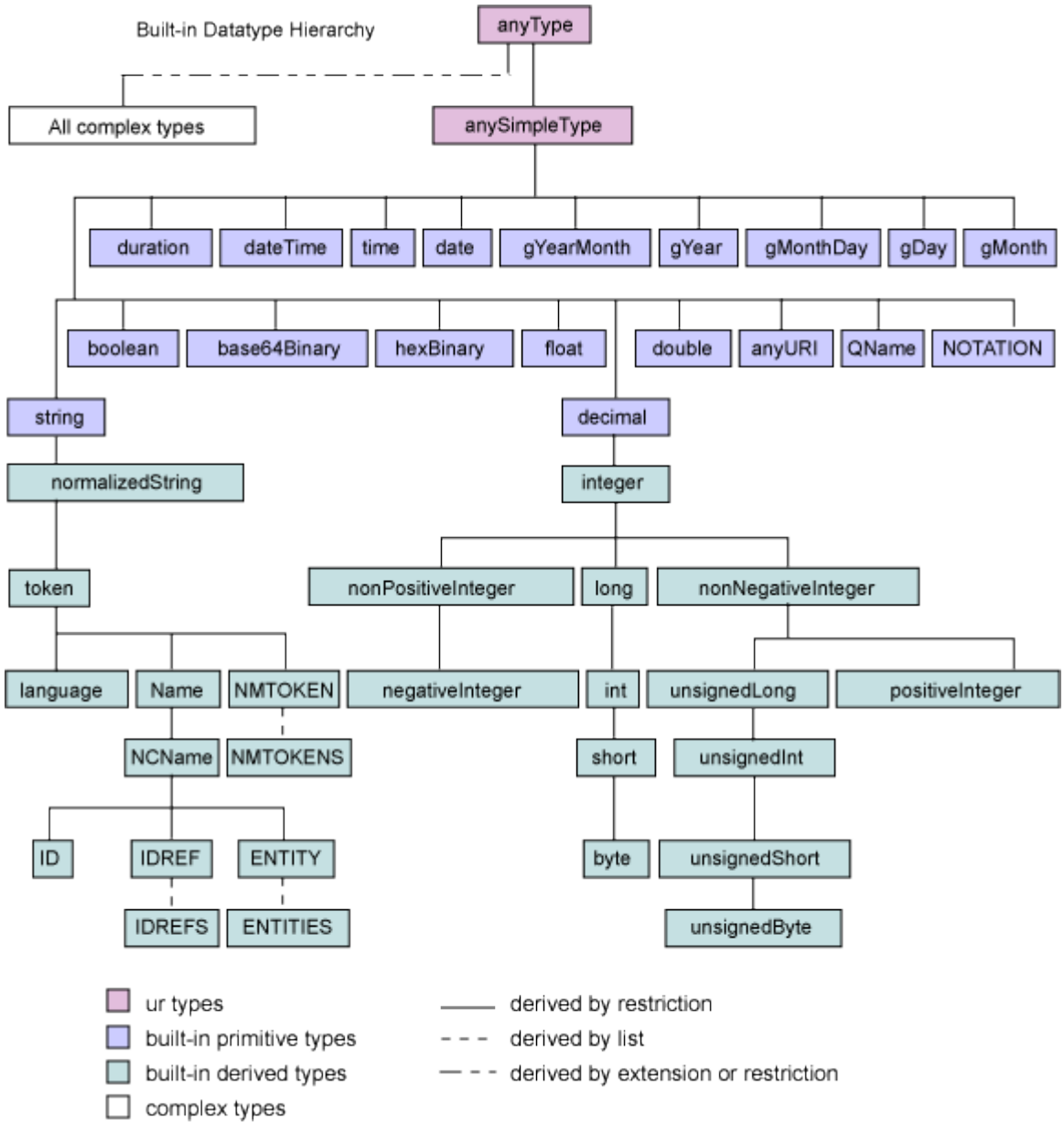
Waarbij UnlimitedNatural staat voor een reeks natuurlijke getallen incl de waarde “unlimited” voorgesteld door een *-teken. Wordt gebruikt door UML zelf voor de weergave van boven- en ondergrens bij multipliciteit.

Bij het modelleren is men in principe vrij om eigen primitieve types te definiëren, maar bij omzetting naar RDFS/OWL vh model door OSLO worden automatisch deze types herkend:

Boolean
Date
DateTime
Double
Duration
HTML
Integer
LangString
Literal
Month
MonthDay
String
Time
URI
Year
YearMonth

Bron: https://github.com/Informatievlaanderen/OSLO-EA-to-RDF

Deze lijst is losweg gebaseerd op de SimpleTypes van de XML Schema Definition Language (XSD):



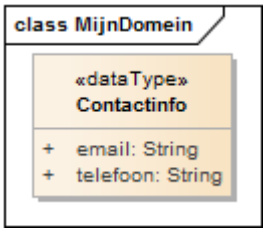
Bron: (W3C, 2004)

REGEL: Gebruik de primitieve datatypes uit bovenstaande lijst en bij uitbreiding deze van XSD.

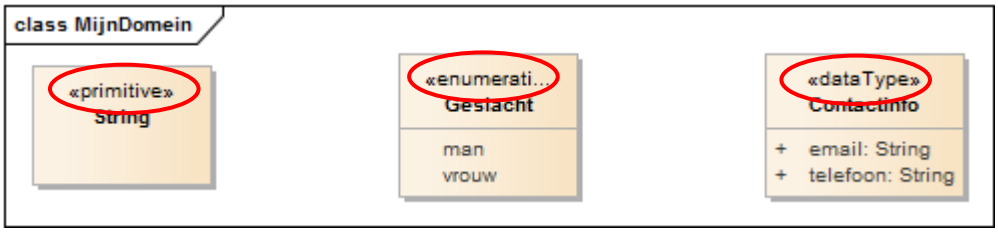
Ook enumeraties zijn een voorstelling vd mogelijke waarden ve attribuut, met dat verschil dat de waarden niet door een wiskundig model worden gegeven maar letterlijk worden opgesomd in een in principe uitputtende lijst, bv:

Opmerking: OSLO breidt het begrip enumeraties uit om ook niet-vlakke lijsten toe te laten (zie §3.2.17).

Gestructureerde datatypes gebruiken we voor waarden die niet atomair zijn en waarvoor verschillende waarden nodig zijn om ze te beschrijven. Ze hebben meerdere attributen (elk op hun beurt ve bepaald datatype), bv:



De weergave van datatypes is identiek aan deze van klassen, met uitzondering vh stereotype boven de naam:

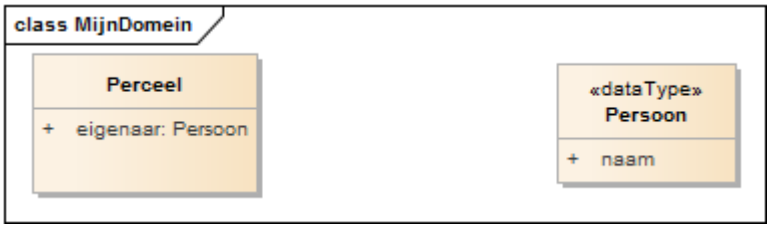


3.2.10 Klasse of datatype?

Zoals gezegd in §3.2.8 is een datatype een classifier voor een verzameling waarden met dezelfde kenmerken. Integers bv zijn een datatype omdat het waarden zijn die voldoen aan eenzelfde wiskundig model.

Instanties ve datatype hebben geen identiteit. De integer “123” gebruikt als waarde van attribuut a in klasse X is identiek aan de integer “123” als waarde van attribuut b in klasse Y. Dat is niet het geval voor instanties van klassen, bv twee instanties “Janssens” ve klasse Persoon hebben niet noodzakelijk betrekking op dezelfde persoon.

Zouden we voor Persoon in voornoemd voorbeeld kiezen voor een datatype, vb:

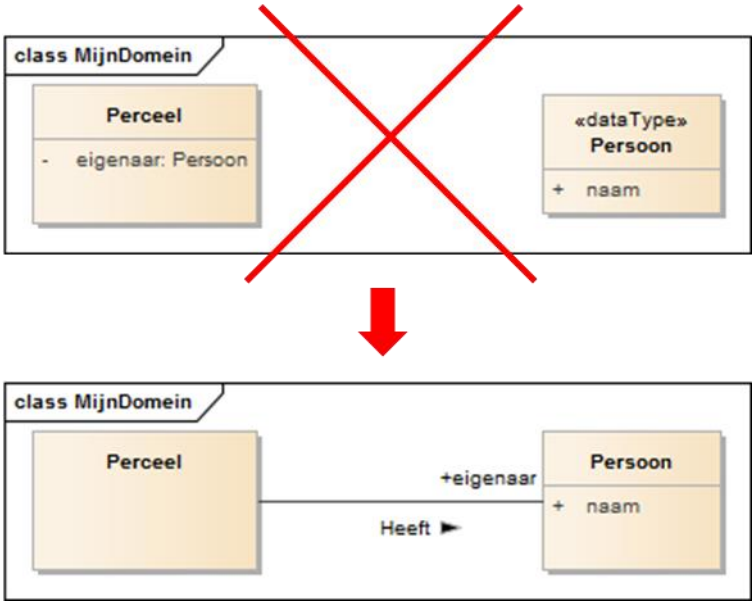


Dan zou dat leiden tot dataredundantie omdat de gegevens van eenzelfde persoon herhaald moeten worden voor elke instantie die naar de persoon verwijst, vb:

```
{
  "Perceel": [
    {
      "-perceelid": "123",
      "eigenaar": {
        "Persoon": {
          "naam": "Janssens"
        }
      }
    },
    {
      "-perceelid": "456",
      "eigenaar": {
        "Persoon": {
          "naam": "Janssens"
        }
      }
    }
  ]
}
```

Omdat de instanties hun identiteit ontlenen aan de waarde die ze aannemen is één schrijffout voldoende om te suggereren dat “Janssens” en “Jansens” verschillende personen zijn. Dit is weliswaar mogelijk, maar kan niet worden geverifieerd omdat de instanties geen identiteit hebben. Men kan dit omzeilen door bv een persoonsnummer op te nemen ih datatype maar dat lost de dataredundantie niet op.

Beter is het om van Persoon dan een klasse maken en die met Perceel te associëren:



Wat leidt tot volgende data:

```
{
  "Perceel": [
    {
      "-perceelid": "123",
      "eigenaar": { "-persoonid": "789" }
    },
    {
      "-perceelid": "456",
      "eigenaar": { "-persoonid": "789" }
    }
  ],
  "Persoon": {
    "-persoonid": "789",
    "naam": "Janssens"
  }
}
```

REGEL: Kies voor een datatype ipv een klasse als instanties vh concept louter door hun waarde worden geïdentificeerd.

Een goed vb van een geval waar we beter wel voor een datatype kiezen is Kost:

Uitleg bij dit vb: Dit is duidelijk een gegeven zonder identiteit, 15Euro is 15Euro, er zijn geen twee 15Euro's.

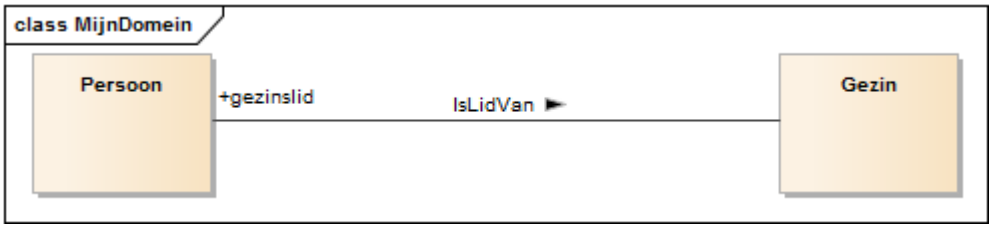
REGEL: Bij twijfel tussen een klasse of een datatype kies voor een klasse.

De kans dat instanties ve klasse alsnog een identiteit blijken te hebben (bv in een andere context) is immers groter en dientengevolge ook de kans op herbruikbaarheid.

3.2.11 Associatierollen, navigatie

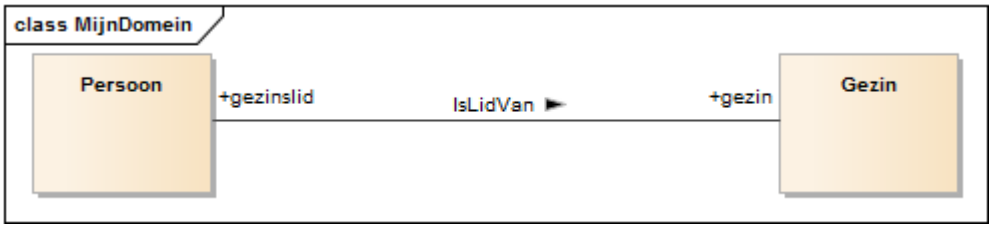
Associaties worden net als klassen en datatypes beschouwd als classifiers, hier classificaties van relaties met gelijke kenmerken (zie (OMG, 2015) p197). Net als een klasse heeft een associatie attributen (zie metamodel §3.2.1), alleen noemen we die hier associatie-uiteinden.

De associatie-uiteinden kunnen optioneel benoemd worden en worden dan rolnamen genoemd. Die geven aan welke rol het uiteinde speelt in de associatie. Bv:

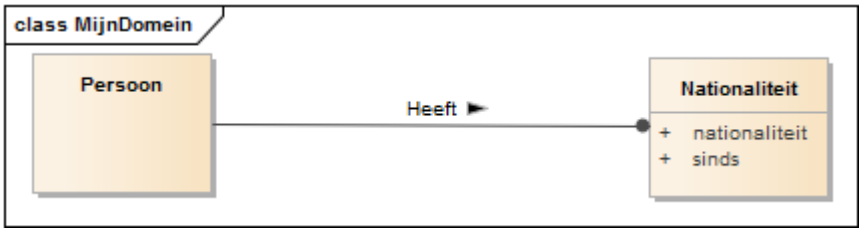


REGEL: Het opgeven van een rolnaam is niet verplicht, doe dit enkel als de rolnaam verschilt van de naam vd geconnecteerde klasse.

Maw: in bovenstaand vb is “gezin” de default rolnaam vh associatie-uiteinde verbonden met Gezin, maw het vb is equivalent met:

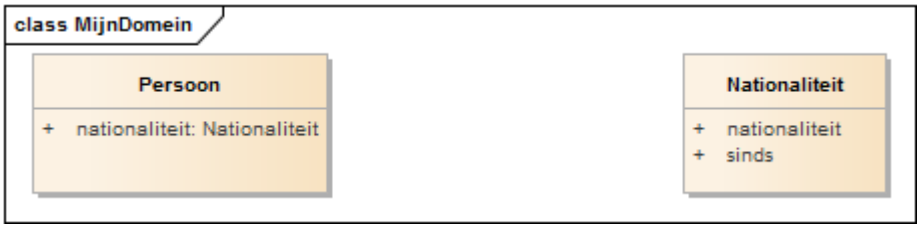


Default is de associatie eigenaar van zijn associatie-uiteinden, wat betekent dat het properties zijn vd associatie (logisch, aangezien een associatie een classifier is, zie hierboven). Echter is het mogelijk om aan te geven dat een verbonden klasse eigenaar is ve associatie-uiteinde, bv:



De dot zegt dat de klasse aan de andere kant vd associatie eigenaar is vh associatie-uiteinde, in het vb is Persoon eigenaar vh associatie-uiteinde met Nationaliteit.

Dit is equivalent met volgende notatie:



Maw: we geven hiermee aan dat de associatie in feite een attribuut is van één van de klassen die ze met elkaar verbindt.

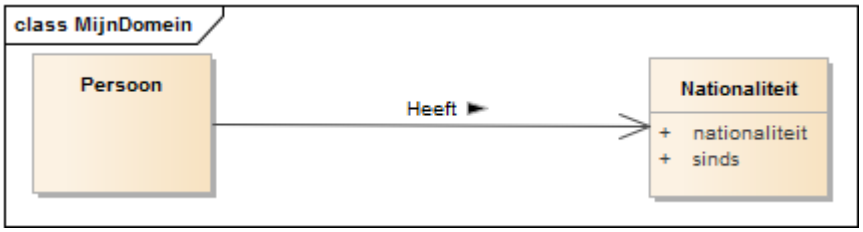
Opmerking: Verschil met een echt attribuut is dat hier een referentie naar de instantie vd doelklasse volstaat (bv zijn identificator). Bij een datatype kan dit niet omdat een instantie daarvan geen identiteit heeft, zie §3.2.10.

REGEL: Om aan te geven dat één vd klassen die de associatie met elkaar verbindt eigenaar is ve associatie-uiteinde, gebruik de notatie met de dot OF maak een attribuut vd associatie.

De keuze wordt bepaald door de leesbaarheid vh diagram maar ook door de overweging in hoeverre de associatie een attribuut is van een klasse ipv een op zichzelf staande entiteit.

Zie §3.2.12 voor bijkomende overwegingen in dit verband.

Opmerking: De notatie met het bolletje vervangt de vroegere informele betekenis vd navigatiepijl in UML, zie (OMG, 2015) p200:



De navigatiepijl geeft aan of vanuit een instantie ve klasse efficiënt de gegevens vd instanties vd geassocieerde klasse opgehaald kunnen worden. Eigenaarschap ve associatie-uiteinde is daar een garantie voor (en de navigatiepijl is dan impliciet) maar kan tegenwoordig ook zonder worden gerealiseerd en is een op zichzelf staand concept geworden.

Sowieso beschouwen we navigatie als iets technisch dat enkel thuishoort in implementatiemodellen.

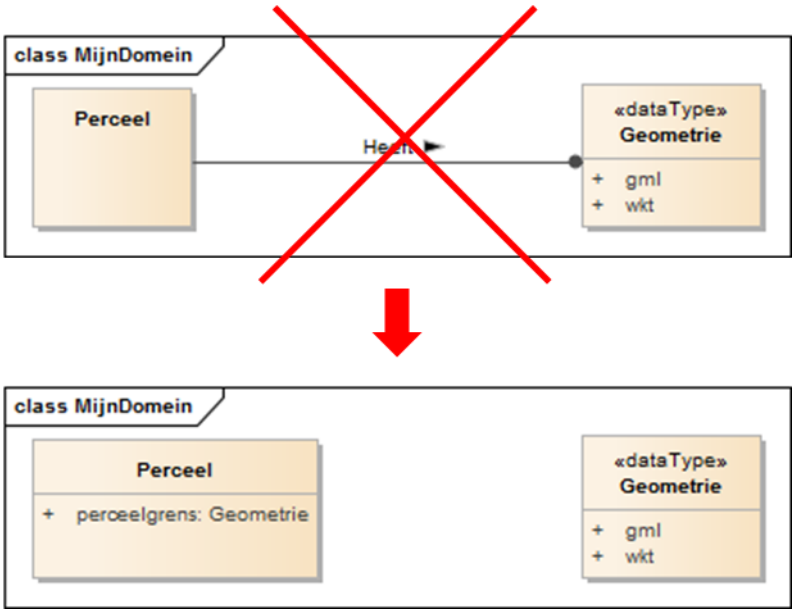
REGEL: Gebruik in conceptuele modellen of applicatiemodellen geen navigatiepijlen op associaties.

3.2.12 Attribuut of associatie?

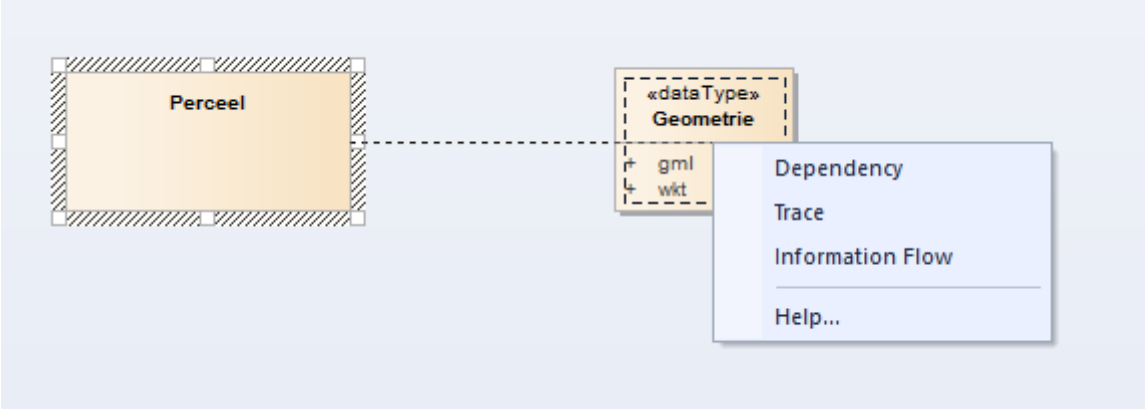
Bij het klasseren van instanties ve concept kan gekozen worden voor een klasse of voor een datatype (zie §3.2.10). Analoog moet bij het modelleren van de kenmerken van klassen of datatypes een keuze worden gemaakt tussen een attribuut of een associatie.

Net als klassen en datatypes is een associatie een classifier (zie metamodel in §3.2.1), ttz een manier om instanties ve concept te klasseren, ih geval ve associatie instanties van relaties. Maw: een associatie dient niet om kenmerken ve klasse of datatype te modelleren, het is een op zichzelf staande entiteit die zelf attributen heeft (mn de associatie-uiteinden die aangeven welke instanties van welke klassen precies door de associatie worden verbonden).

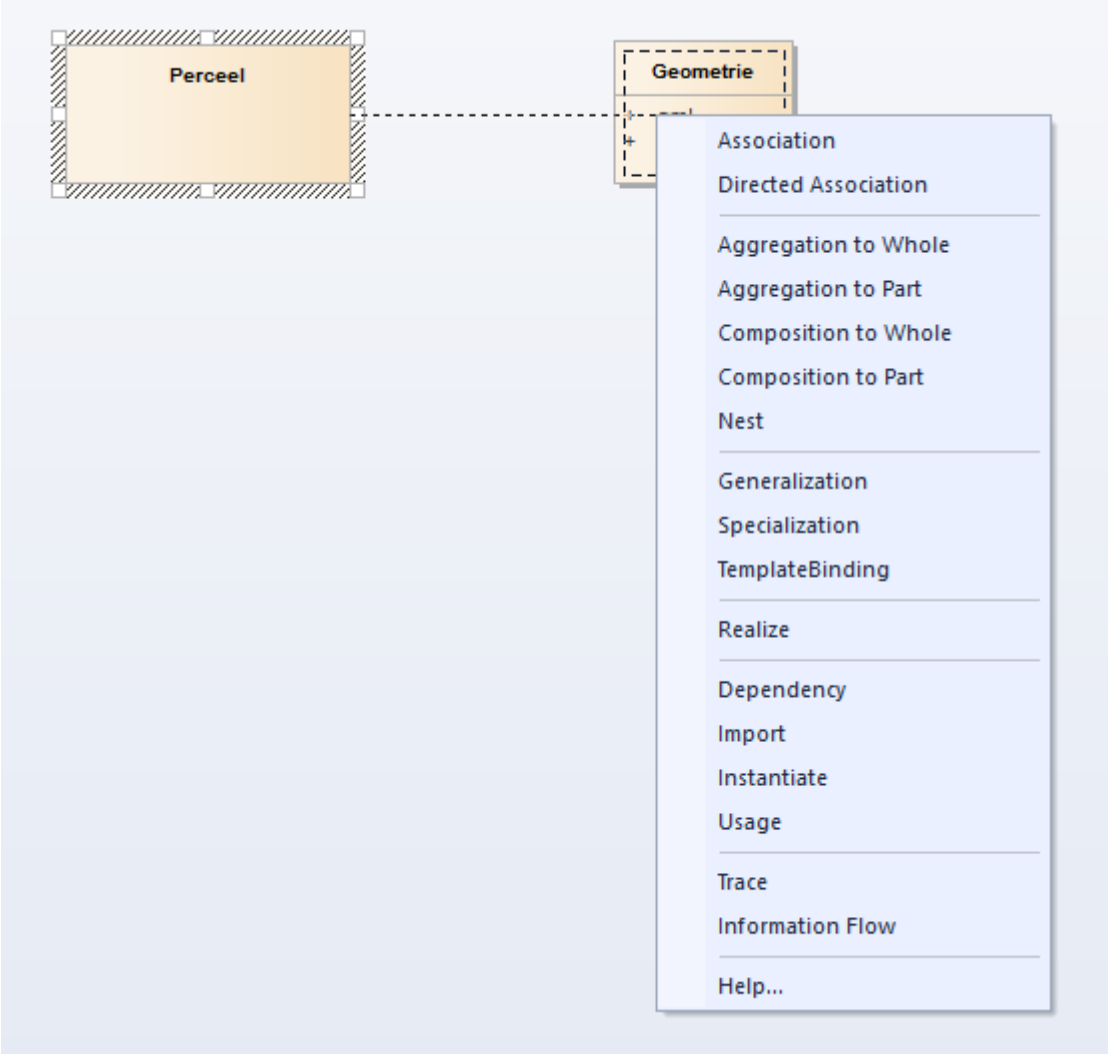
In theorie kan het uiteinde ve associatie ook een datatype zijn, in dat geval echter is een associatie niet mogelijk, vb:



Hoewel het metamodel alle types classifier lijken toe te laten als type ve associatie-uiteinde (zelfs associaties?) blokkeren modelleringstools als Enterprise Architect (SPARX) de mogelijkheid om een klasse te associëren met een datatype:

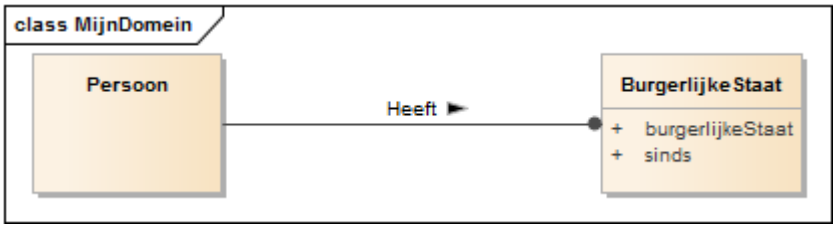


Vergelijk met de opties wanneer we willen associëren met een klasse:

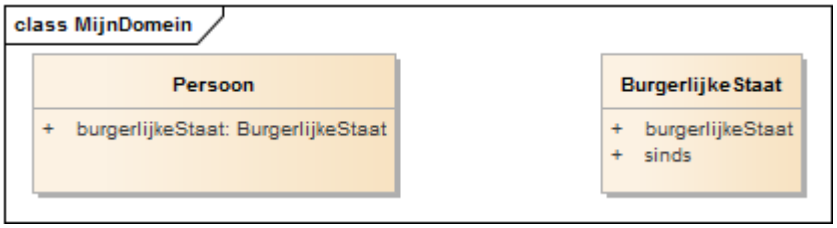


Alvast zou een associatie met een datatype nooit de vorm ve referentie kunnen aannemen (mogelijkheid beschreven in §3.2.11) aangezien instanties van datatypes geen identiteit hebben.

Het probleem stelt zich dus enkel als het associatie-uiteinde vd classifier aan het einde vd associatie een klasse is, en verder ook alleen als de klasse aan het begin vd associatie eigenaar is vh overeenkomstige associatie-uiteinde, zie §3.2.11. Bv:



Uitleg bij dit vb: Zegt dat Persoon eigenaar is vh associatie-uiteinde dat verwijst naar BurgerlijkeStaat. Equivalent is in dat geval deze notatie:



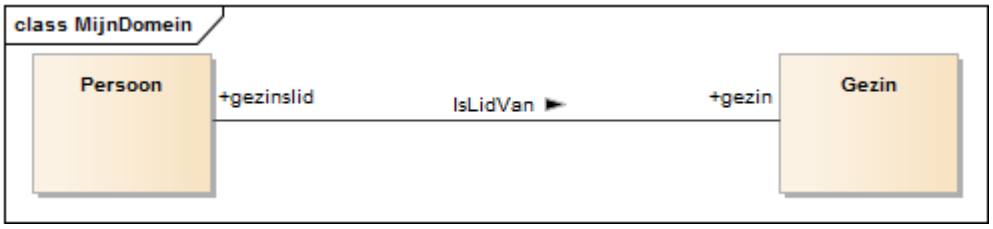
In §3.2.11 zeiden we dat de keuze voor associatie of attribuut bepaald wordt door leesbaarheid vh diagram en de mate waarin het associatie-uiteinde een kenmerk is vd klasse. Maar dit is eerder vaag, daarom volgende vuistregel:

REGEL: Is de classifier aan het einde vd associatie een datatype maak dan een attribuut, is de classifier een klasse creëer dan een associatie.

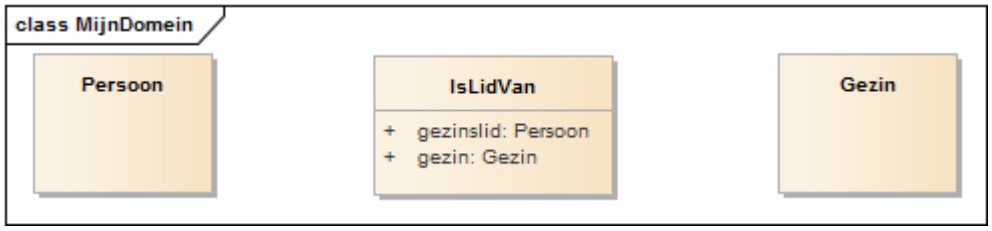
UML legt deze conventie niet op maar suggereert ze wel, zie (OMG, 2015) §9.5.3.

3.2.13 Associatieklassen

In het metamodel (zie §3.2.1) zagen we dat associaties eigenlijk classifiers zijn, maw het is een soort klasse met associatie-uiteinden als attributen. Bv dit:



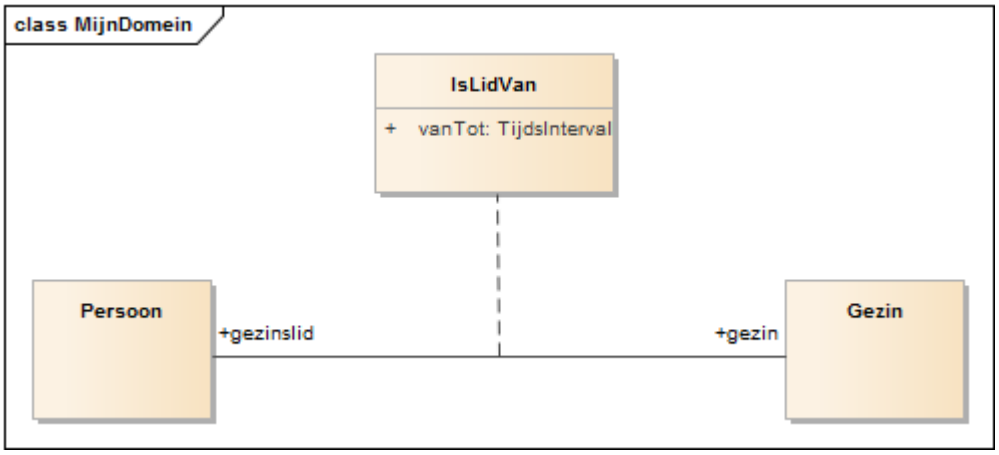
Is eigenlijk equivalent aan:



Opmerking: Deze notatie is louter ter illustratie, ze is in praktijk ongewenst (zie §3.2.7).

Het komt voor dat behalve de associatie-uiteinden meer attributen nodig zijn om de associatie te beschrijven, bv:

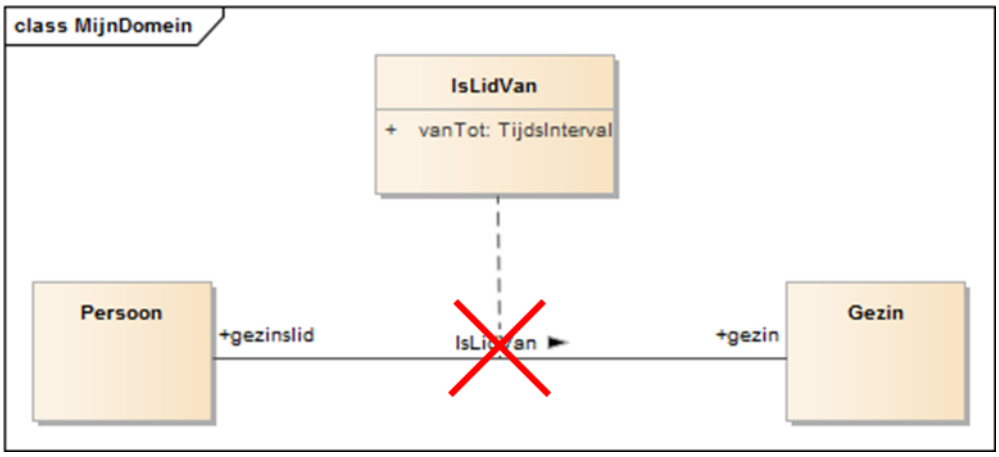
Voor een dergelijk geval introduceert UML een zgn associatieklasse, di een classifier die zowel klasse als associatie is (zie metamodel §3.2.1). Notatie van bovengenoemd vb is dan als volgt:



REGEL: Als een associatie attributen heeft, creëer dan een associatieklasse.

Volgens het metamodel (zie §3.2.1) kan een associatieklasse geen datatype zijn, het is een classifier van associaties, niet van waarden.

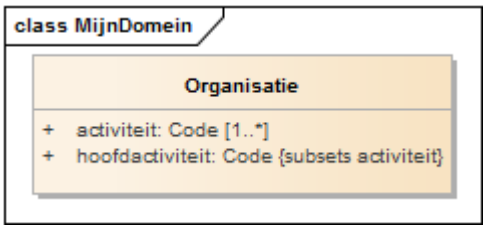
Zoals gezegd is een associatieklasse zowel een associatie als een klasse. Dat betekent echter niet dat het door meervoudige overerving twee namen heeft, zie (OMG, 2015) p198. Maw de associatienaam in onderstaand model is overbodig:



3.2.14 Specialisatie van structurele kenmerken

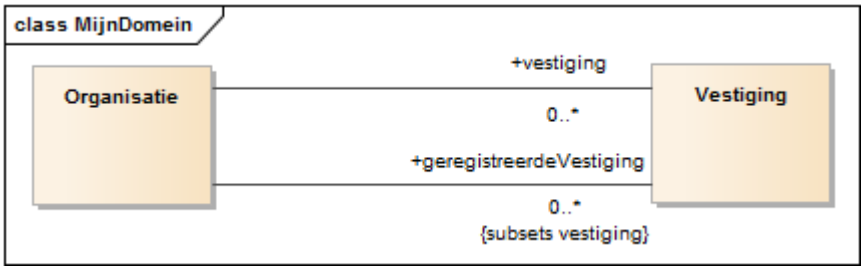
UML biedt de mogelijkheid om attributen en associaties te specialiseren. Dit gebeurt door de modifier “subsets” toe te voegen.

Vb specialisatie ve attribuut:



Uitleg bij dit vb: Een Organisatie kan meerdere activiteiten uitoefenen, waarvan er één beschouwd kan worden als de hoofdactiviteit.

Vb specialisatie ve associatie:



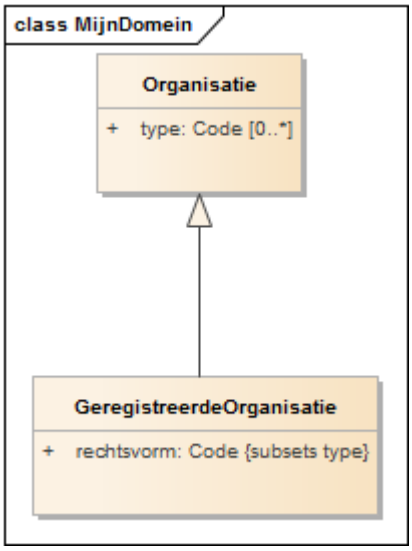
Uitleg bij dit vb: Een Organisatie kan geassocieerd zijn met meerdere vestigingen, een deel van die associaties slaat op vestigingen die officieel geregistreerd zijn bv omdat de organisatie er vergunningsplichtige activiteiten uitoefent.

Opmerking: Andere mogelijkheid in bovenstaand vb was het creëren ve subklasse GeregistreerdeVestiging. Echter: als meerdere organisaties gebruik kunnen maken van eenzelfde vestiging, waarbij deze voor sommige Organisaties geregistreerd is en voor sommige niet is dat geen oplossing.

REGEL: Plaats de modifier “subsets” bij een rolnaam als niet de klasse maar de associatie met die klasse gespecialiseerd moet worden.

Opmerking 2: Het is niet echt de associatie die wordt gespecialiseerd, eerder zijn associatie-uiteinden (ook wel rollen genoemd). Dit laat toe om bv het ene uiteinde wel en het andere niet te specialiseren, zoals het geval in bovenstaand voorbeeld.

Soms is het nodig om in een subklasse het kenmerk vd superklasse te specialiseren en ook dat is mogelijk in UML, bv:



Uitleg bij dit vb: Er zijn verschillende opties om organisaties te klasseren, GeregistreerdeOrganisaties echter worden doorgaans door hun juridisch statuut getypeerd (NV, VZW, Stad/Gemeente etc).

3.2.15 Multipliciteit

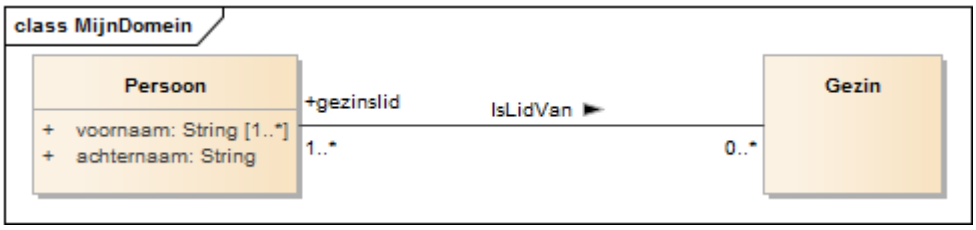
Multipliciteit in een UML-klassendiagram geeft de kardinaliteit weer van een concept. Kardinaliteit staat voor het aantal objecten dat een collectie bevat, hier het potentieel aantal objecten (omdat we niet de werkelijkheid voorstellen maar een model vd werkelijkheid).

De multipliciteit bestaat in principe uit twee niet-negatieve integers waarbij de eerste de ondergrens vd kardinaliteit geeft en de tweede de bovengrens, bv 0..1 betekent dat een concept niet voorkomt of maximaal 1 keer. Voor de bovengrens mag echter ook een sterretje worden opgegeven, dit om aan te geven dat het aantal onbegrensd is, bv 1..* betekent dat het aantal 1 of meer is.

Multipliciteit kan enkel voor structurele kenmerken, ttz voor bv attributen en associatie-uiteinden en niet voor behavioral kenmerken zoals bv operaties (zie figuur in §3.2.2 voor de opdeling structural/behavioral). In het geval van attributen worden in het klassendiagram vierkante haakjes geplaatst rond de multipliciteit, bv [0..*].

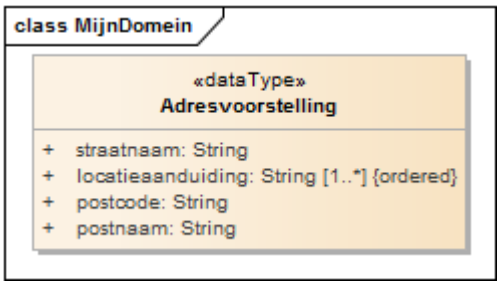
Wordt geen multipliciteit getoond dan is deze 1..1.

Voorbeeld:



Uitleg bij dit vb: De multipliciteiten op de attributen van Persoon betekenen dat een persoon ten minste één voornaam moet hebben en steeds één achternaam. Ook op de associatie-uiteinden vd associatie IsLidVan staan kardinaliteiten en die duiden aan dat een gezin ten minste één gezinslid heeft en dat het mogelijk is dat een persoon geen deel uitmaakt van een gezin of deel uitmaakt van één gezin of zelfs meerdere gezinnen.

Default zijn de waarden in de verzameling waarop de multipliciteit betrekking heeft uniek en ongeordend. Het is echter mogelijk in UML om dit aan te passen. Vb:



Uitleg bij dit vb: Een Adres kan meerdere locatieaanduidingen hebben bv huisnummer 12 en bus 1 die in een Adresvoorstelling liefst in een bepaalde volgorde worden geplaatst, maw (huisnummer) 12 bus 1 en niet bus1 (huisnummer) 12.

Zoals al gezegd in §3.1.1:

REGEL: Vermeld kardinaliteiten van concepten enkel in applicatieprofielen.

Het opnemen van kardinaliteiten (en andere constraints, zie §3.2.16) in een vocabularium wordt strijdig geacht met de Open World Assumption (OWA), al eerder vermeld in §2.3, zie (INSPIRE, 2017) §9.4.7. Volgens OWA is kennis nooit volledig en kunnen we bv niet met zekerheid stellen dat een persoon altijd een achternaam heeft.

Applicatieprofielen gaan uit vd Closed World Assumption en dat we dit wel weten, toch ten minste binnen de applicatie waarin we de concepten uit het vocabularium worden toegepast (om bij het vb van de achternaam te blijven: in een Belgisch personenregister wordt steeds een achternaam geregistreerd).

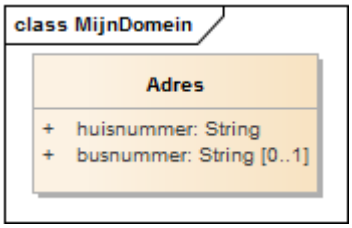
OSLO houdt enkel rekening met semantische kardinaliteit, praktische kardinaliteit wordt genegeerd. Het feit dat een gegeven weliswaar van toepassing is binnen het domein of de toepassing maar desnoods kan of mag ontbreken bij data-uitwisseling (omdat het niet gekend is of niet is ingewonnen of niet mag worden meegedeeld) wordt niet opgenomen in het model.

REGEL: Modelleer enkel de semantische kardinaliteit.

Maw: een ondergrens 0 in de multipliciteit is mogelijk, maar niet een stereotype zoals <voidable> bij INSPIRE, zie (INSPIRE, 2008).

Opmerking: Begrippen als verplicht/optioneel vallen onder praktische kardinaliteit.

Vb van semantische kardinaliteit met ondergrens 0:



Uitleg bij dit vb: Een adres heeft steeds een huisnummer maar niet altijd een busnummer.

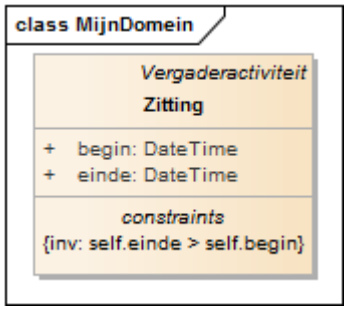
REGEL: Is een structureel kenmerk niet relevant voor de toepassing, zet dan niet de ondergrens vd multipliciteit op 0 maar verwijder het kenmerk uit het model.

Bv iedereen heeft een geboortedatum maar in sommige toepassingen is deze niet nodig.

3.2.16 Constraints

Onder constraints verstaan we voorwaarden, beperkingen of beweringen gerelateerd aan de elementen in het model. In die zin verschillen ze niet van de semantiek vh domein die al in het model vervat zit. Verschil is dat het hier gaat om semantiek die niet dmv de grafische syntax van UML kan worden voorgesteld.

Vb:



Uitleg bij dit vb: Een vergadering kan niet eindigen voor ze begonnen is, hier uitgedrukt dmv een OCL-expressie die waar is als het tijdstip waarop de vergadering eindigt groter is dan het tijdstip waarop ze begon.

Niet alle constraints horen thuis in een OSLO-model:

REGEL: Beperk de constraints tot businessregels, voeg geen technische constraints toe aan het model.

Vb:

Uitleg bij dit vb: Deze constraint zegt dat een huisnummer niet langer dan 50 karakters kan zijn. Echter bestaat daarover geen businessregel in het domein van adresregistratie, deze regel is ingegeven door de technische implementatie van het model bv in een databank waar men een maximum lengte wil geven aan de string die dit gegeven voorstelt.

Zoals al gezegd in §3.1.1:

REGEL: Vermeld constraints op concepten enkel in applicatieprofielen.

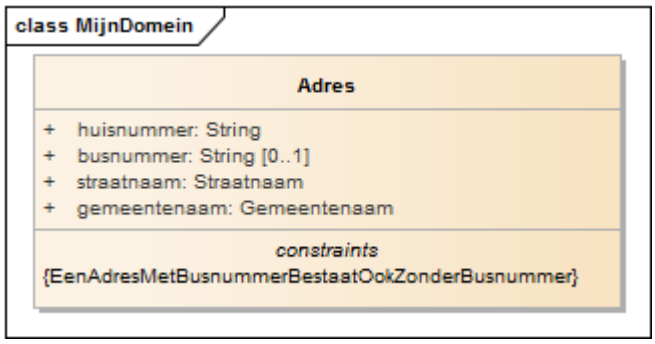
Constraints in een vocabularium worden strijdig geacht met de Open World Assumption (OWA), meer info in §3.2.15.

UML legt niet op in welke taal constraints moet geschreven worden, dat kan in natuurlijke taal of in een meer formele taal zoals OCL. OCL is de afkorting van Object Constraint Language en is net als UML ontwikkeld door de OMG, zie (OMG, 2014).

OSLO maakt platformonafhankelijke modellen, gebruik dus geen programmeertaal als JAVA om constraints te beschrijven.

REGEL: Definieer constraints indien mogelijk in OCL en anders in natuurlijke taal.

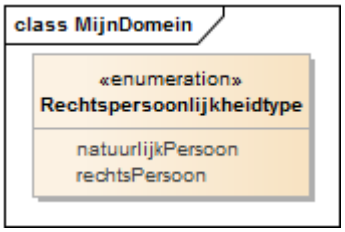
Bv:



Uitleg bij dit vb: De constraint wordt hier beschreven in natuurlijke taal, het zou gespecialiseerde kennis vergen om hem in OCL uit te drukken (als dit zelfs al mogelijk is).

3.2.17 Enumeraties

Enumeraties zijn gespecialiseerde datatypes. Net als andere datatypes geven ze de mogelijke waarden die een attribuut kan aannemen, deze waarden zijn echter letterlijke waarden ipv waarden gegeven door een wiskundig model. Vb:

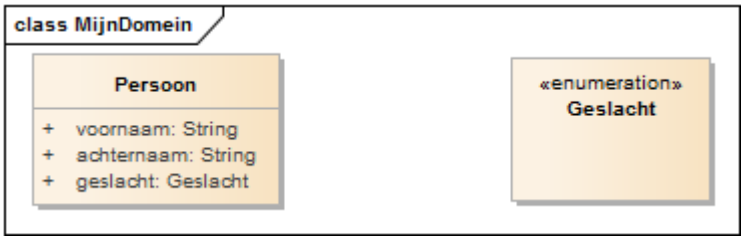


Voor uitwisseling van data is een opsomming van mogelijke waarden genoeg. Echter: OSLO-modellen focussen op semantiek, ttz op de betekenis van de concepten achter de waarden. In bovenstaand vb: wat is een NatuurlijkPersoon, welke relaties bestaan er tussen dit begrip en andere begrippen zoals Gezin of Domicilie? Een eenvoudig lijstje kan die semantiek niet beschrijven, er is een apart model nodig per enumeratie.

REGEL: Maak een model voor elke enumeratie of verwijs naar bestaande enumeratiemodellen.

De verwijzing naar een enumeratiemodel gebeurt door een tag “ap-codelist” toe te voegen aan de enumeratie of aan het attribuut dat de enumeratie als datatype heeft. Meer info in §3.2.19 over metadata.

Het is dus niet nodig om in het klassendiagram in de enumeraties die erin voorkomen waarden te zetten, de enumeraties mogen leeg zijn, bv:



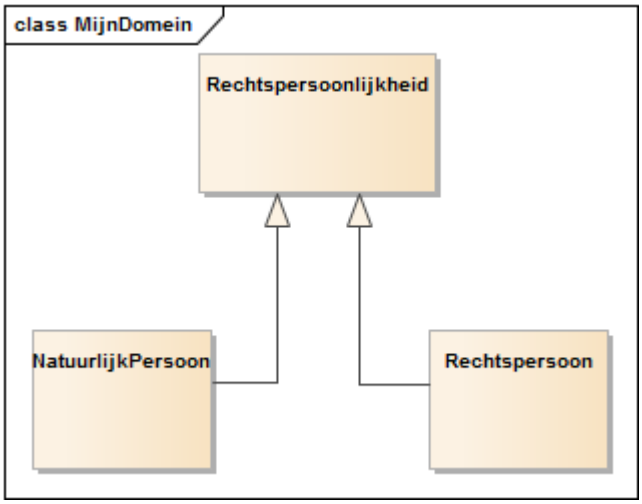
Verder geldt:

REGEL: Verwijs enkel naar enumeratiemodellen in een AP.

Uitgewerkte enumeraties in een vocabularium worden strijdig geacht met de Open World Assumption (OWA), meer info in §3.2.15.

Vraag is hoe zo een enumeratiemodel er moet uitzien.

Een UML-klassendiagram zoals we dat gebruiken om onze OSLO-modellen te beschrijven (aangevuld met metadata zoals definities vd concepten) zou perfect zijn, bv:



Echter: in de meeste gevallen gaat het louter om relaties tussen de begrippen, typisch hiërarchische relaties (het enumeratiemodel is dan een taxonomie) soms aangevuld met associatieve of equivalente relaties (het enumeratiemodel is dan een thesaurus). Dus geen attributen, zoals naam en voornaam van een NatuurlijkPersoon in bovenstaand vb.

REGEL: Beschrijf eigen enumeratiemodellen in SKOS.

Het semantisch web ontwikkelde speciaal een VOC om lijsten/taxonomieën/thesauri te beschrijven: SKOS of Simple Knowledge Organization System, zie (W3C). In tegenstelling tot UML is de syntax tekstueel vb:

```
@prefix dct:       <http://purl.org/dc/terms/> .
@prefix rdfs:      <http://www.w3.org/2000/01/rdf-schema#> .
<https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid>
       a skos:ConceptScheme ;
       dct:modified "2018-07-23" ;
       skos:definition "Rechtspersoonlijkheid betekent dat men een rechtssubject is, dwz drager
       van rechten en plichten. Natuurlijke personen hebben vanzelf rechtspersoonlijkheid,
       rechtspersonen (bvb venootschappen) moeten die van rechtswege toegekend krijgen. "@nl ;
       skos:prefLabel "Rechtspersoonlijkheid"@nl .
<https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid/NatuurlijkPersoon>
       a skos:Concept ;
       skos:prefLabel "Natuurlijk Persoon"@nl ;
       skos:definition "De rechtspersoonlijkheid is deze ve natuurlijk persoon."@nl ;
       skos:broader <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> ;
       skos:topConceptOf <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> ;
       skos:inScheme <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> .
<https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid/Rechtspersoon>
       a skos:Concept ;
```

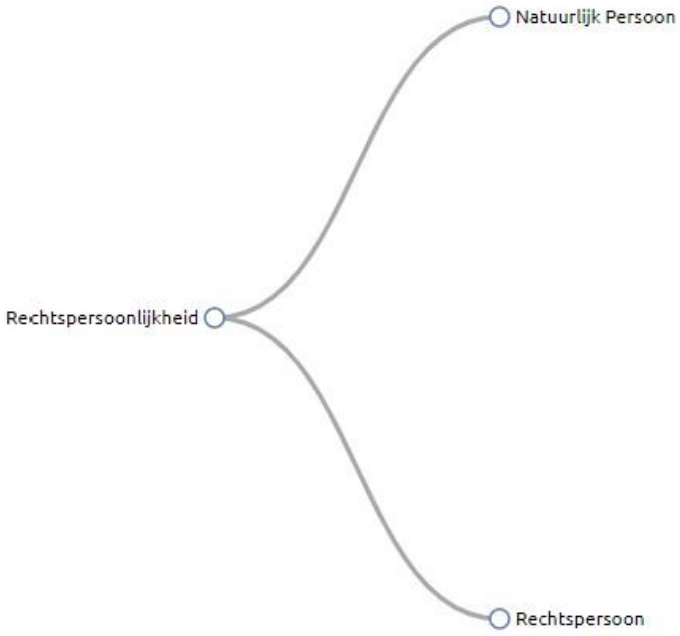
```
skos:prefLabel "Rechtspersoon"@nl ;
skos:definition "De rechtspersoonlijkheid is deze ve rechtspersoon."@nl ;
skos:broader <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> ;
skos:topConceptOf <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> ;
skos:inScheme <https://data.vlaanderen.be/id/conceptscheme/Rechtspersoonlijkheid> .
```

In dit vb zien we enkele belangrijke elementen vd SKOS-syntax:

- Conceptscheme: beschrijft de enumeratie
- Concept: beschrijft een concept dat voorkomt in de enumeratie
- Broader: geeft aan dat het concept een specialisatie is ve ander concept

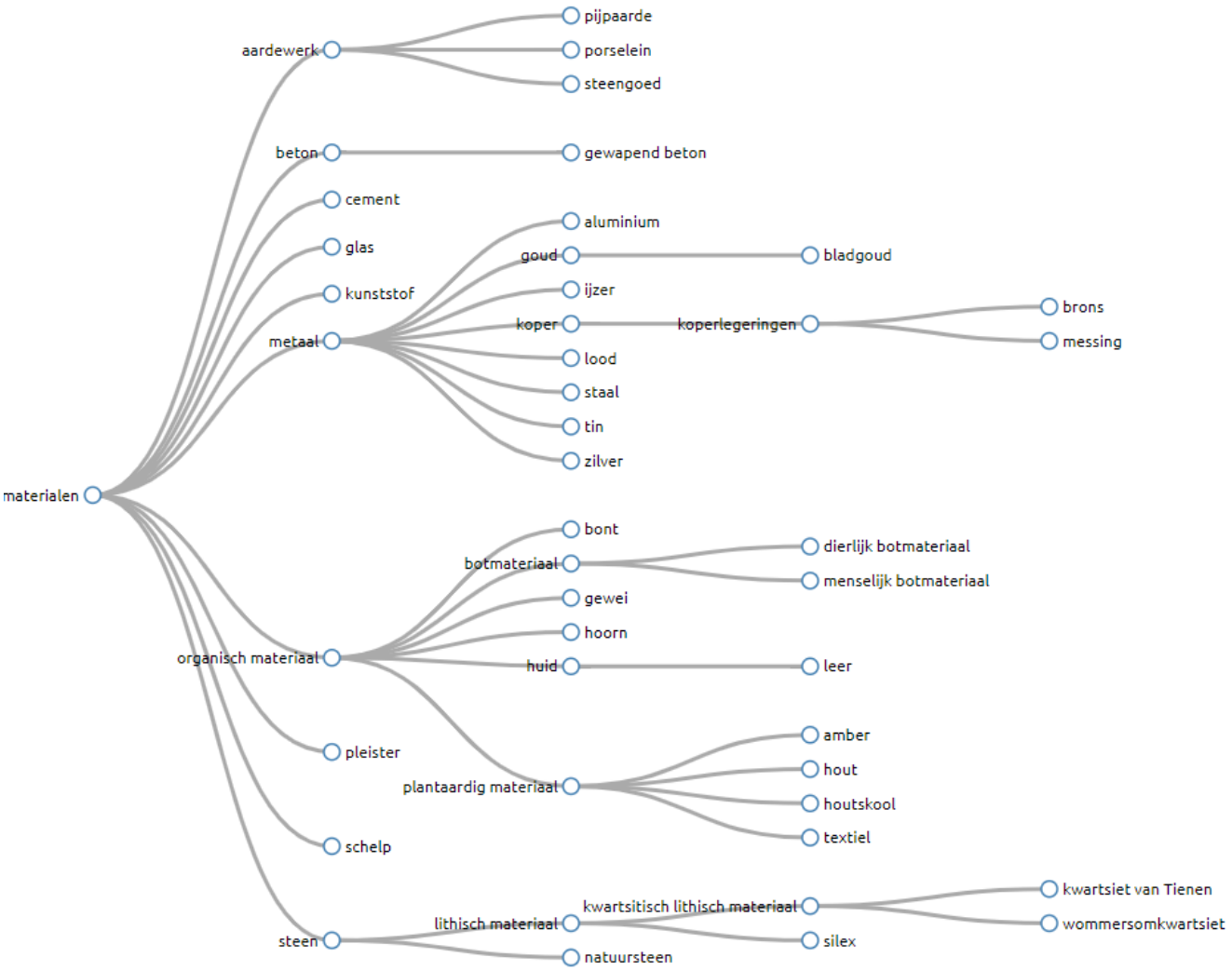
Gelijkaardig aan broader zijn narrower & related, maw met deze termen uit het SKOS-vocabularium worden de semantische relaties tussen de concepten gemodelleerd. Bemerk de analogie met de zgn mapping termen (ook uit SKOS) vermeld in §3.1.2).

De grafische voorstelling maakt geen deel uit vd syntax maar visualisatie dmv een boom is gebruikelijk:



Gevisualiseerd met SKOS Play.

Dit vb is wel heel elementair, vergelijk met deze meer complexe structuur:



Bron: https://thesaurus.onroerenderfgoed.be/conceptschemes/MATERIALEN.

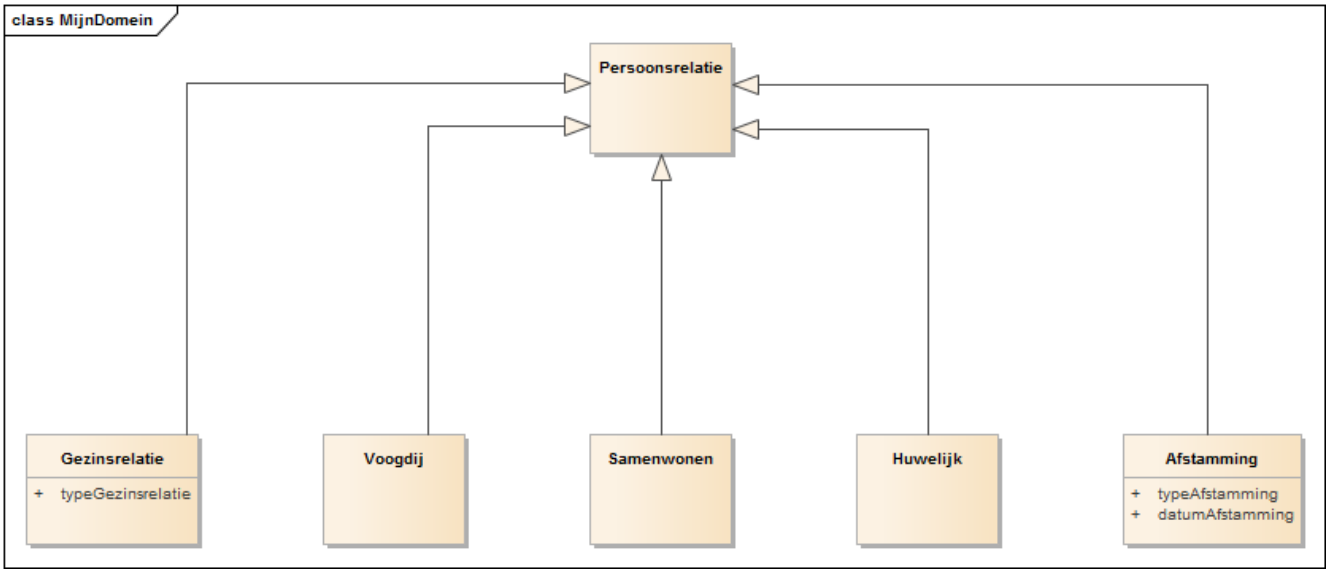
Opmerking: Bestaande enumeratiemodellen zullen niet altijd in SKOS gemodelleerd zijn. Verwijs in dat geval naar de documentatie waarin de enumeratie wordt beschreven.

3.2.18 Subklassen of enumeratie?

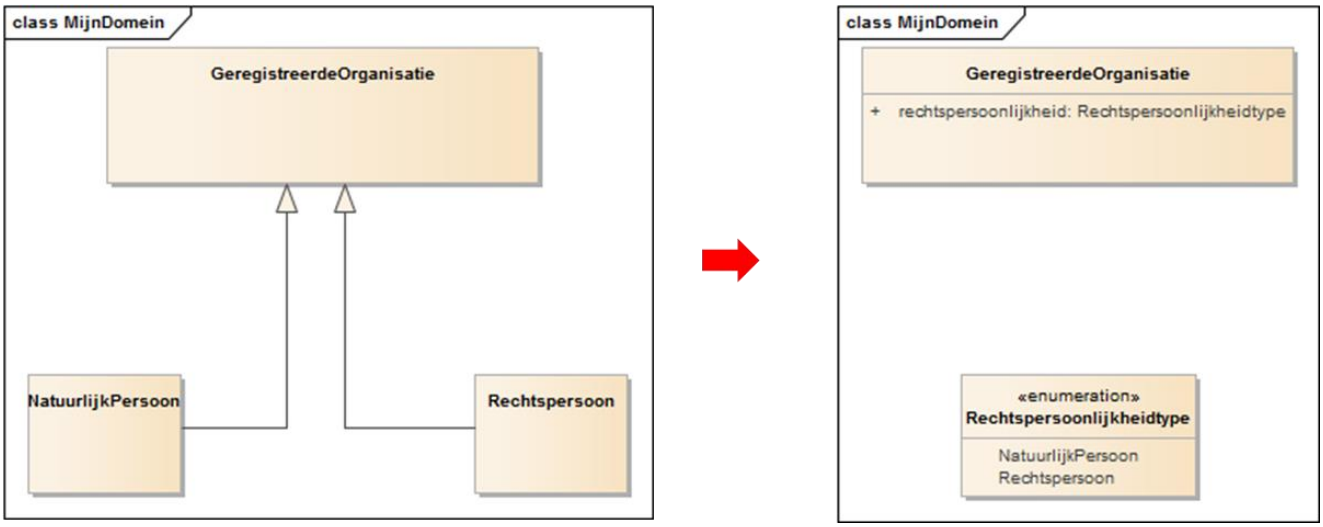
Het is niet altijd nodig om volledige classificaties van concepten te modelleren dmv subklassen.

REGEL: Als één of meerdere mogelijke subklassen eigen attributen of associaties hebben die de superklasse niet heeft, specialiseer dan de superklasse.

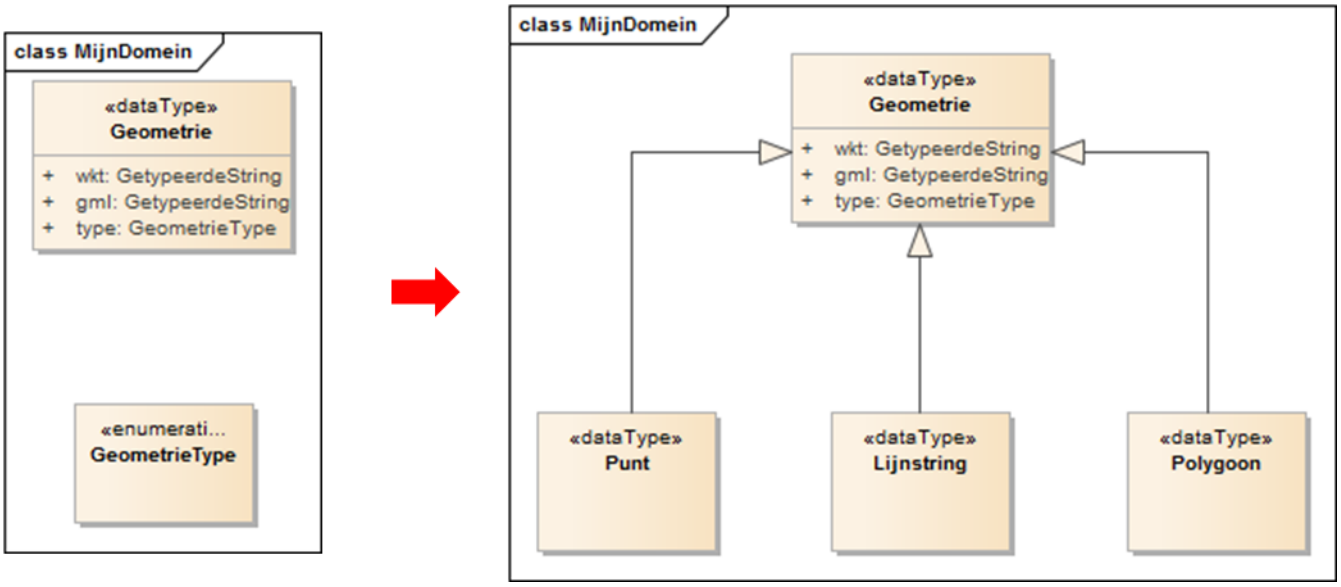
Bv:



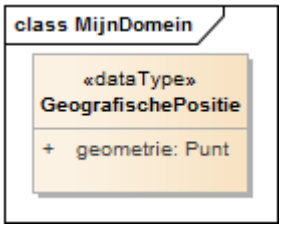
Is deze regel niet van toepassing gebruik dan een enumeratie (zie §3.2.17), vb:



Toch zijn er gevallen waarin subklassen beter zijn, bv:



Uitleg bij dit vb: Een subklasse maakt het type geometrie zichtbaar in het model, bv als volgt:



3.2.19 Metadata

REGEL: Voorzie elk concept in het model van volgende metadata:

- label
- definition (indien geen bestaand concept bij VOC)
- usageNote (indien geen bestaand concept bij VOC)
- uri (indien bestaand concept bij VOC)

Opmerking: UML voorziet het mechanisme vd “tagged values” om dergelijke metadata bij te houden. Het OSLO UML-profiel voorziet in stereotypes waaraan deze tags al zijn toegevoegd, zie

Annex A: UML-profiel.

Wat de betekenis vd tags betreft worden volgende afspraken gemaakt:

Tag	Toelichting
label	Mensleesbare naam vh concept, bv Publieke Organisatie als de identificator vh concept ih diagram PubliekeOrganisatie is. Deze naam wordt gebruikt in de documentatie bij het schema.
definition	Samenvattende omschrijving vh concept.
usageNote	Aanvullende omschrijving, bv met voorbeelden.
uri	URI vh concept.

Aan de naam vd tag wordt nog een taalcode toegevoegd om aan te geven in welke taal de waarde vd tag is gesteld, bv definition-nl.

De naam vd tag wordt voorafgegaan door de prefix ap als de tag betrekking heeft op het AP ipv op het VOC, bv ap-definition-nl.

Op de naam vd uri-tag zijn deze overwegingen mbt prefix/postfix niet van toepassing.

Definitie en Gebruiksnota hoeven niet te worden gegeven indien het een concept uit een bestaand VOC betreft. In dat geval kunnen definitie en evt gebruiksnota vh concept immers worden opgehaald via de uri. In het geval ve AP moet steeds een definitie en gebruiksnota worden gegeven als deze afwijkt van de definitie of gebruiksnota vh overeenkomstig concept in het VOC.

Vb PubliekeOrganisatie:

Op modelniveau geldt volgende regel:

REGEL: Voorzie het model van volgende metadata:

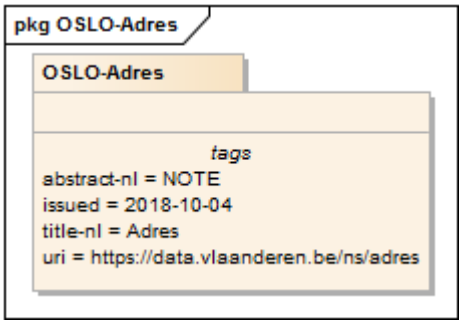
- title
- abstract
- issued
- uri

Wat de betekenis vd tags betreft worden volgende afspraken gemaakt:

Tag	Toelichting
title	Titel vh model.
abstract	Korte omschrijving vh model.
issued	Publicatiedatum.
uri	URI vh model.

Aan de naam vd tag wordt nog een taalcode toegevoegd om aan te geven in welke taal de waarde vd tag is gesteld, bv definition-nl.

Vb OSLO-Adres:



Voor een enumeraties geldt:

REGEL: Voorzie een enumeratie van volgende metadata:

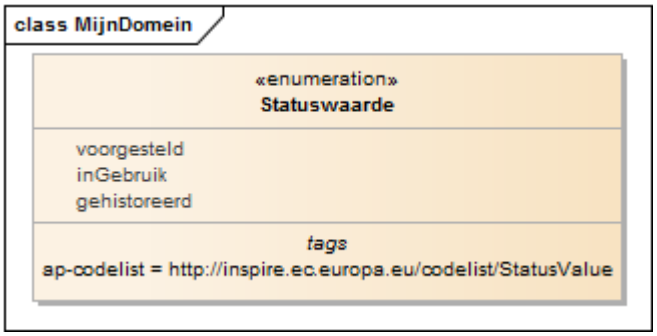
- ap-codelist

Wat de betekenis vd tags betreft worden volgende afspraken gemaakt:

Tag	Toelichting
ap-codelist	URI met verwijzing naar het enumeratiemodel.

De waarde vd tag ap-codelist is typisch een URI van een dmv SKOS beschreven enumeratiemodel, maar het kan ook een URI zijn naar een document waarin dat model beschreven wordt.

Vb:



Opmerking: De enumeratiewaarden in bovenstaand vb zijn louter ter informatie, de werkelijke opsomming is beschreven in het enumeratiemodel waarnaar de uri verwijst.

Opmerking 2: Postfix “ap” maakt sowieso deel uit van de naam vd tag omdat enumeraties enkel in AP’s worden uitgewerkt, ttz enkel in AP’s wordt verwezen naar het model dat de enumeratie beschrijft. Meer info in §3.2.17.

3.2.20 Naamgeving

Voor conventies op het vlak van naamgeving richten we ons in grote lijnen op de ISO (zie (ISO 19103, 2015) p 35).

De naam ve concept is identificerend binnen het bereik vh element waartoe het behoort. Bv: de naam ve attribuut is uniek binnen zijn klasse, een rolnaam is uniek voor zijn associatie, een klasse is uniek binnen zijn package etc.

REGEL: Zorg dat de naam ve element uniek is tov zijn parent in het metamodel.

Bv:

Omdat de naam ve concept dikwijls uit meerdere woorden bestaat en geen blanco’s of andere scheidingstekens in de naam mogen voorkomen geldt volgende afspraak:

REGEL: Schrijf samengestelde woorden en zinnen aan elkaar.

Maw: blanco’s of andere scheidingstekens zijn niet toegelaten.

Bv:

Opmerking: Samenstelling zijn in het Nederlands in principe al aaneen geschreven, bv straatnaam. In het Engels is dat niet zo, bv street name.

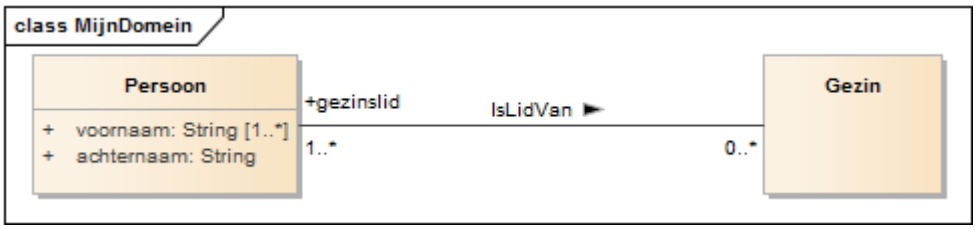
Om de woorden toch nog van elkaar te kunnen onderscheiden passen we CamelCase toe.

REGEL: Gebruik UpperCamelCase voor Klassen, Datatypes, Enumeraties, Associaties en Associatieklassen, Primitieven.

Maw: voor alles wat een classifier is (zie metamodel §3.2.1). Voor elementen die een kenmerk ve classifier zijn gebruiken we lowerCamelCase:

REGEL: Gebruik lowerCamelCase voor attributen, rolnamen, enumeratiewaarden, operaties.

Vb:



Met het oog op de leesbaarheid voegen we voor namen in het Nederlands nog volgende regel toe:

REGEL: Samenstellingen in het Nederlands worden bij het toepassen van CamelCase beschouwd als één woord.

Dus niet ContactInfo maar Contactinfo, niet achterNaam maar achternaam.

Zie §3.2.7 voor bijkomende conventies bij de naamgeving van associaties.

3.2.21 Defenties

REGEL: Zorg dat elk concept dat voorkomt in het model rechtstreeks of onrechtstreeks gedefinieerd is.

In praktijk betekent dit dat er een definitie moet zijn voor alle Klassen, Attributen, Datatypes, Associaties, Enumeraties en Enumeratiewaarden.

Een rechtstreekse definitie zit bij het model, gegeven via de definition-tag. Een onrechtstreekse definitie is mogelijk via de uri-tag vh concept. Zie §3.2.19 over metadata voor meer info.

REGEL: Gebruik geen circulaire beschrijvingen.

Bv niet: een PubliekeOrganisatie is een organisatie die publiek is.

REGEL: Begin een definitie niet met het herhalen vd naam vh concept.

Bv voor Adres: “Een Adres is informatie die toelaat om…” moet zijn “Informatie die toelaat om…”.

REGEL: Een definitie is een samenvattende omschrijving, zorg dat ze geen voorbeelden of bijkomende uitleg bevat.

Aanvullende omschrijvingen worden gegeven in de tag usageNote, zie §3.2.19.

De definitie ve concept uit een VOC kan verschillen van hetzelfde concept in een AP:

REGEL: In een AP verfijn indien nodig de definitie vh overeenkomstig concept uit een VOC.

Dit is dikwijls het geval als een AP een bestaand VOC herbruikt of als het eigen VOC waarop het AP zich baseert zelf termen uit bestaande VOC’s herbruikt.

Bv definitie van Organisatie.beschrijving in de VOC OSLO-Organisatie en in het AP OSLO-OrganisatieBasis:

VOC	AP
Beschrijving vd resource.	Beschrijving van de organisatie.

Het concept “beschrijving” is overgenomen van http://purl.org/dc/terms/description uit het DCTERMS-vocabularium en krijgt in het AP OSLO-OrganisatieBasis een meer gespecialiseerde toepassing (beschrijving vd organisatie ipv de resource).

3.2.22 Zichtbaarheid

UML biedt de mogelijkheid om het gebruik van elementen te controleren door de zichtbaarheid ervan te beperken. Mogelijke waarden:

- Public
- Private
- Protected
- Package

UML heeft zijn wortels in OO-programmeren en een belangrijk principe daar is het verbergen van attributen en operaties die met implementatie te maken hebben. Een attribuut met zichtbaarheid private zal niet geraadpleegd kunnen worden door een programma met toegang tot het object.

Deze notie van informatie verbergen is relevant voor implementatiemodellen maar niet voor de conceptuele - en applicatiemodellen waarop OSLO focust.

REGEL: Zet de zichtbaarheid van elementen steeds op “public”.

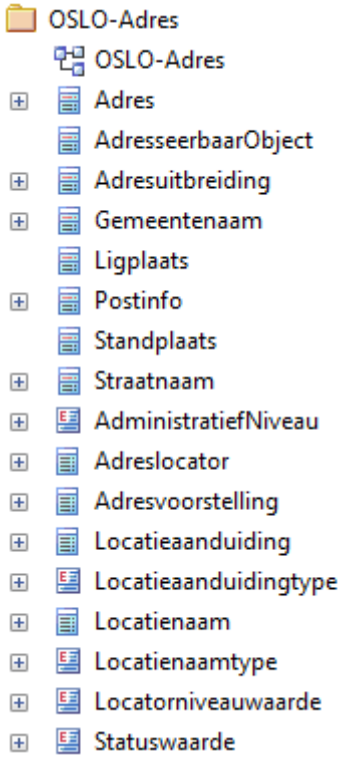
Semantische datastandaarden bevatten louter termen die relevant zijn voor het beschreven domein en dus zetten we de zichtbaarheid ervan op public.

3.2.23 Pakketten

Een VOC of AP bestaat uit een model en een schema, zie §2.4. Het model is de collectie van de elementen waaruit het bestaat, het schema de formele voorstelling van het model.

Een pakket (package in het Engels) is een constructie die toelaat de elementen ve model te groeperen, incl het schema:

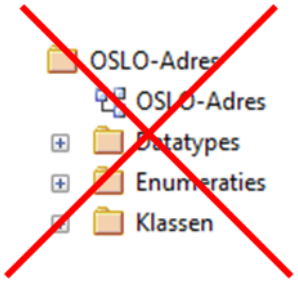
Vb:



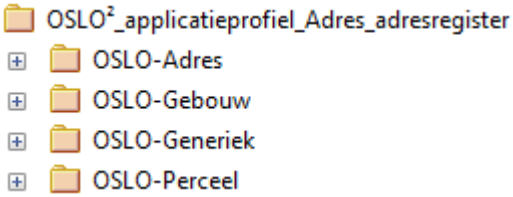
REGEL: Creëer een pakket voor een model en zijn schema.

Voeg de vereiste metadata toe aan dit pakket, zie §3.2.19 ivm metadata. Dit pakket vertegenwoordigt ahw het model en zijn schema (en dus het domein dat het model beschrijft).

De elementen kunnen gegroepeerd worden in subpakketten. Vermijd daarbij een technische indeling zoals:



Soms worden termen herbruikt uit andere VOC's of AP's. Let er op om deze bij uitwisseling mee te exporteren, groepeer evt de benodigde pakketten in een bovenliggend pakket, bv:



Uitleg bij dit vb: Het AP OSLO-Adresregister maakt gebruik van elementen uit het VOC OSLO-Adres, OSLO-Gebouw, OSLO-Generiek en OSLO-Perceel.

3.3 REGELS MBT DIAGRAMLAYOUT

De regels:

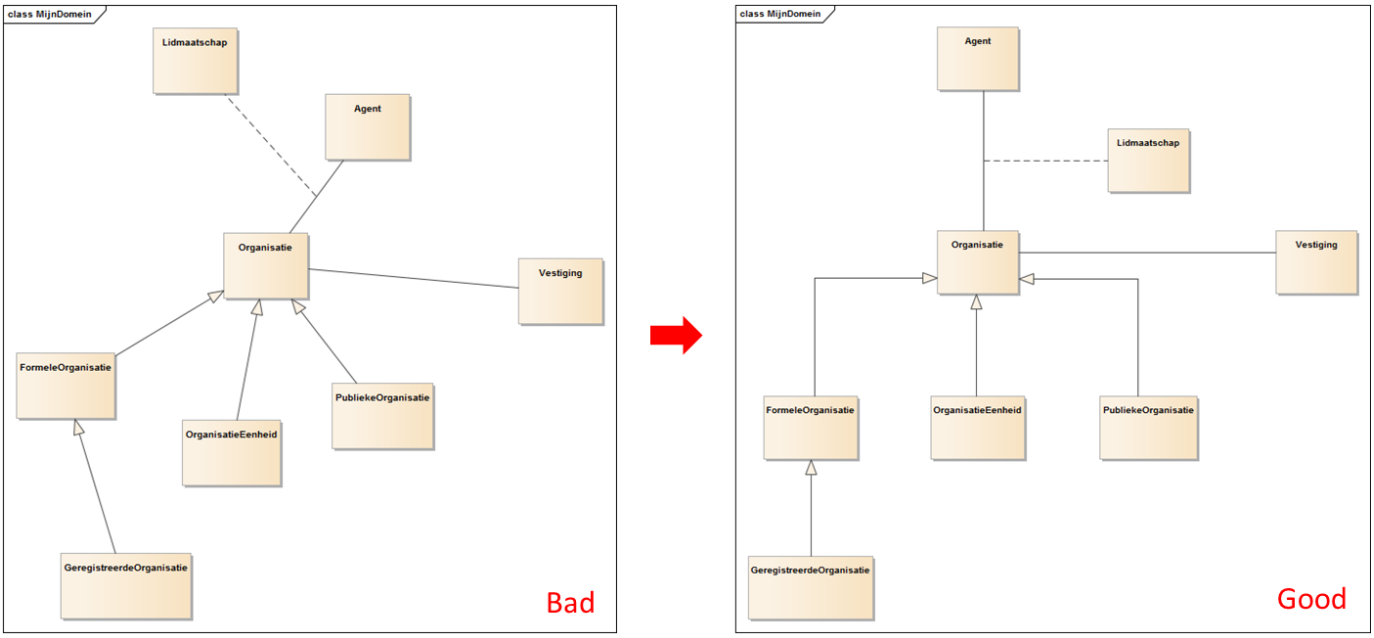
- Orthogonaliteit.
- Afmetingen harmoniseren.
- Less is more.
- Geen kruisingen.
- Parents bovenaan.

Deze sectie is gebaseerd op (Bellekens, 2012) welke werd overgenomen door (ISO TC211).

3.3.1 Orthogonaliteit

REGEL: Aligneer elementen horizontaal & verticaal en gebruik rechte of haakse connectoren.

Dit wordt geïllustreerd door volgend vb:

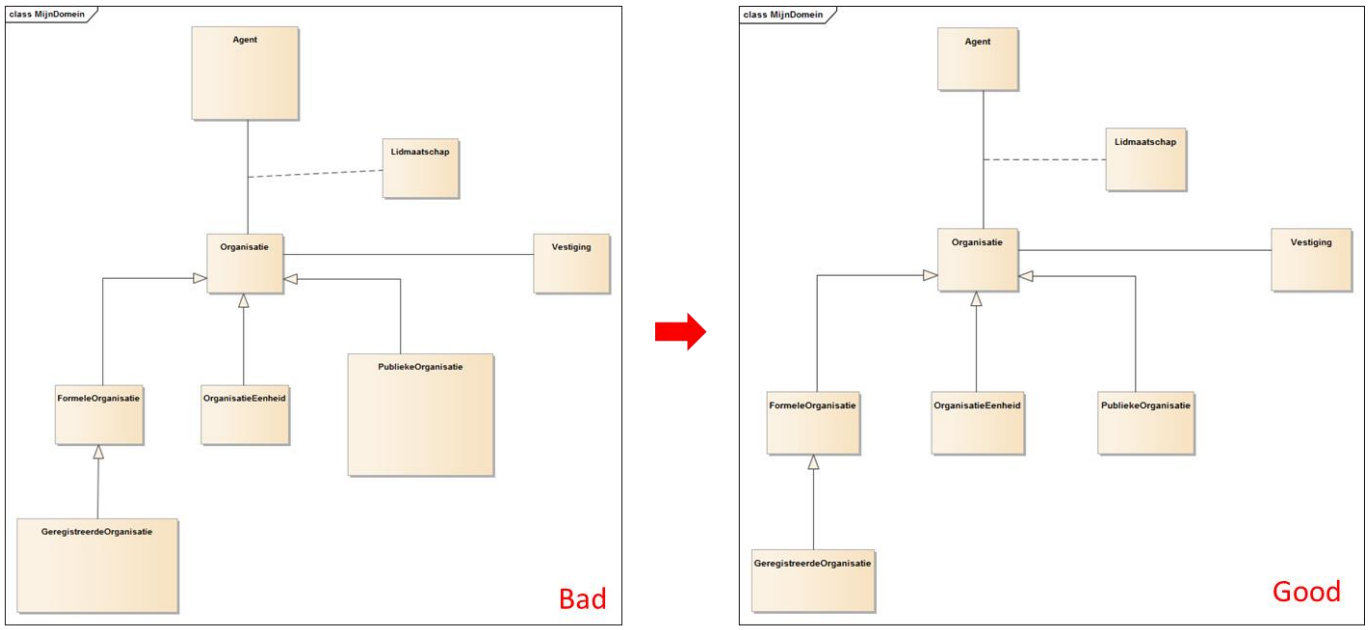


Dit levert een minder rommelig en dus meer leesbaar diagram op.

3.3.2 Afmetingen harmoniseren

REGEL: Maak elementen van gelijk belang even groot.

In onderstaand vb wordt in het linkse diagram gesuggereerd dat een publieke organisatie belangrijker zou zijn dan een gewone organisatie.



3.3.3 Less is more

Hoe minder elementen er voorkomen op een diagram, hoe leesbaarder het wordt.

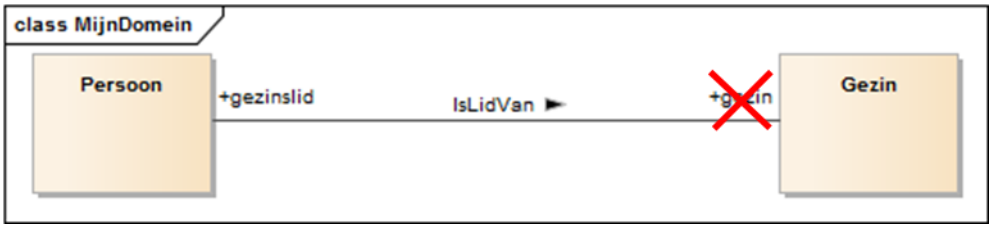
REGEL: Vereenvoudig een complex diagram door:

- Het op te splitsen in een aantal kleinere diagrammen.
- Door bepaalde stukken ve diagram uit te lichten.
- Door attributen, rolnamen ed die niet relevant zijn te verbergen.

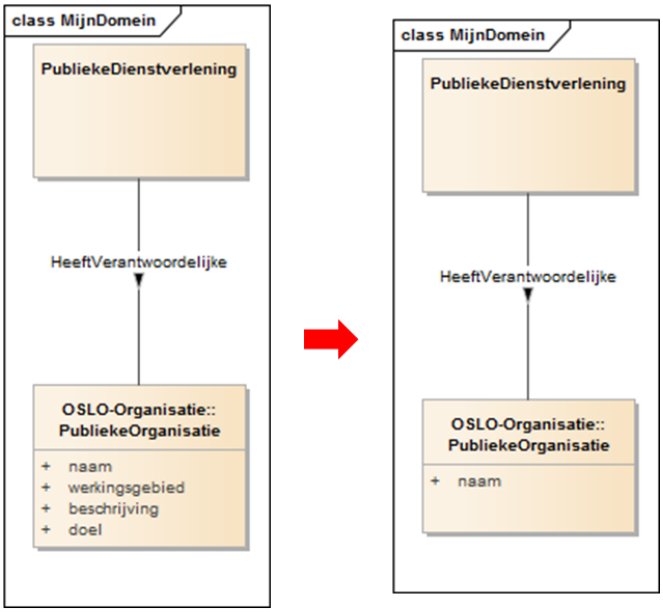
De eerste van deze remedies past in de goede praktijk van modulair ontwerp, zie §0.

De tweede komt erop neer om de documentatie ve diagram op te splitsen en daarbij telkens maar een deel vh diagram te tonen, evt verwijzend naar een overzichtsdiagram dat op zijn beurt vereenvoudigd is door bv attributen te verbergen.

De derde oplossing heeft oa betrekking op rolnamen die niet expliciet moeten worden vermeld, bv:



Of op attributen van klassen uit een VOC die ih kader ve AP niet relevant zijn en dus verborgen mogen worden, bv:

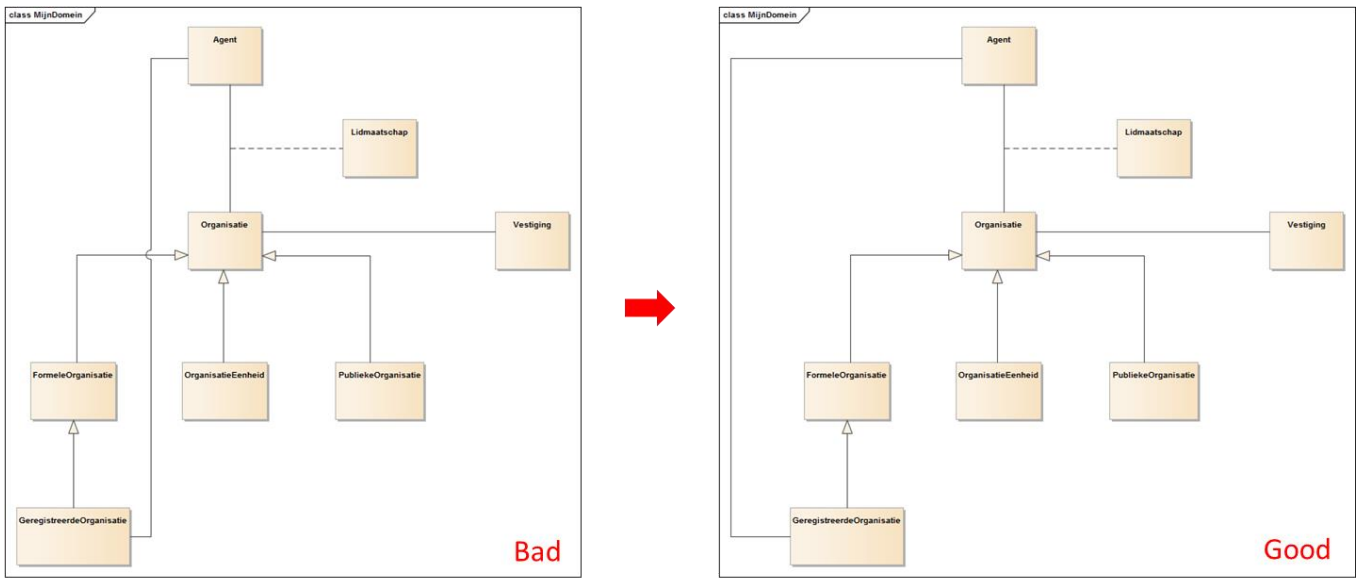


Bad Good

3.3.4 Geen kruisingen

REGEL: Vermijd kruisende connectoren.

Vb:

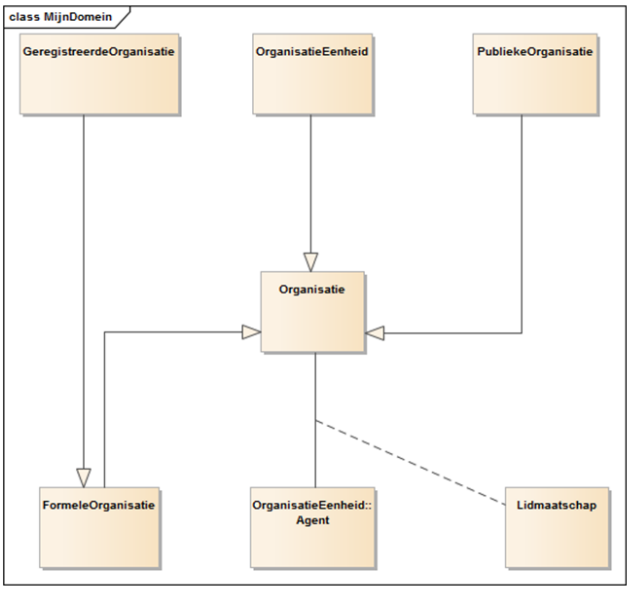


3.3.5 Parents bovenaan

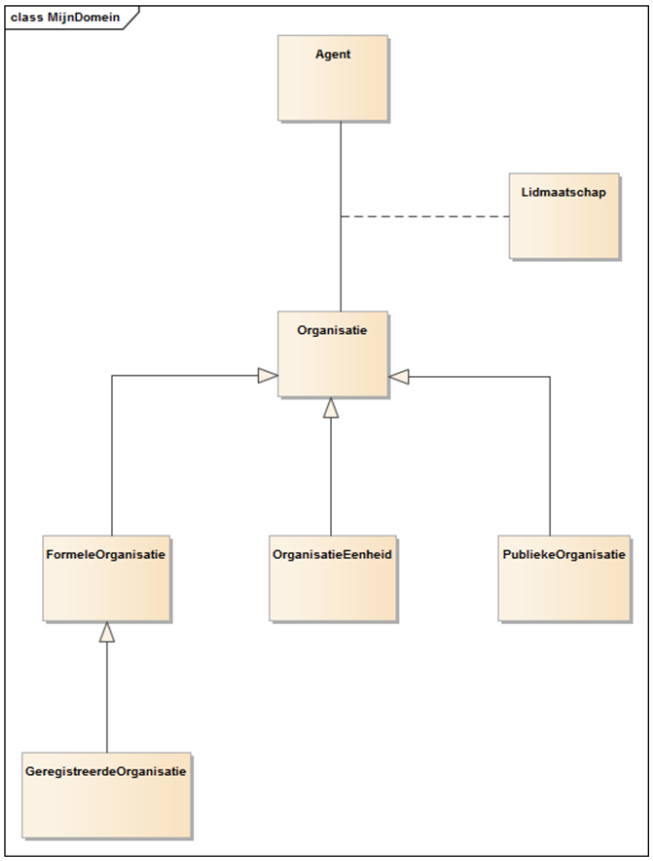
Classificaties worden doorgaans van boven naar onder of van links naar rechts gelezen, ttz van het meer algemene naar het meer specifieke. Daarom:

REGEL: Plaats subklassen onder of rechts van hun parent en plaats bij aggregaten of composieten het onderdeel onder of rechts van het geheel.

Vb:



Bad



Good

4 REFERENTIES

AWV. (sd). *Modelleringshandleiding OTL.*

Bellekens. (2012). *UML Best Practice: 5 rules for better UML diagrams.*

Colomb. (2006). *The Object Management Group Ontology Definition Metamodel.* Opgehaald van https://pdfs.semanticscholar.org/27a0/1e2aa277def52ee6e3599f1e53aa7b324a48.pdf

Connors. (2009). *From Taxonomies to Ontologies.* Opgehaald van https://www.slideshare.net/TriviumRLG/from-taxonomies-to-ontologies

Denemarken. (sd). *Rules for Concept and Data Modelling.*

Fakhroutdinov, K. (sd). *UML-Diagrams.* Opgehaald van https://www.uml-diagrams.org

Informatie Vlaanderen. (2018). *Proces en methode voor het ontwikkelen van datastandaarden.*

INSPIRE. (2008). *Generic Conceptual Model.*

INSPIRE. (2017). *Guidelines for the RDF encoding of spatial data.* Opgehaald van http://inspire-eu-rdf.github.io/inspire-rdf-guidelines/#ref_cr_prop_multiplicity

ISA. (sd). *Handbook for using Core Vocabularies.* Opgehaald van https://joinup.ec.europa.eu/site/core_vocabularies/Core_Vocabularies_user_handbook/ISA%20Handbook%20for%20using%20Core%20Vocabularies.pdf

ISO 19103. (2015). *Conceptual Schema Language.*

ISO 19109. (2015). *Geographic information — Rules for application schema.*

ISO 19110. (sd). *Methodology for feature cataloguing.*

ISO TC211. (sd). *UML-Best-Practices.* Opgehaald van https://github.com/ISO-TC211/UML-Best-Practices/wiki

Kriouile. (sd). *An MDA Method For Automatic Transformation of Models from CIM to PIM.* Opgehaald van https://pdfs.semanticscholar.org/31f9/124d2f1d353b5b4414b41fcdf0c3fa1f93a7.pdf?_ga=2.217722262.1869965235.1528732449-95705632.1528732449

OGC. (sd). *OWS-8 Domain Modelling Cookbook.* Opgehaald van https://github.com/ISO-TC211/UML-Best-Practices/blob/master/Reference%20material/11-107_OWS-8_Domain_Modelling_Cookbook.pdf

OMG. (2014). *Model Driven Architecture (MDA), MDA Guide rev 2.0.* Opgehaald van https://www.omg.org/cgi-bin/doc?ormsc/14-06-01

OMG. (2014). *Object Constraint Language.* Opgehaald van https://www.omg.org/spec/OCL/About-OCL/

OMG. (2014). *Ontology Definition Metamodel.* Opgehaald van file:///C:/Users/thijsge/Downloads/formal-14-09-02.pdf

OMG. (2015). *UML Specification version 2.5.* Opgehaald van https://www.omg.org/spec/UML/2.5.1/PDF

Snodgrass, R. T. (2000). *Developing Time-Oriented Database Applications in SQL.*

SPARX. (sd). *UML metamodel xmi.*

W3C. (1998). *A Discussion of the Relationship Between RDF-Schema and UML.* Opgehaald van https://www.w3.org/TR/1998/NOTE-rdf-uml-19980804/

W3C. (2004). *XML Schema Part 2: Datatypes Second edition.*

W3C. (2014). *Best Practices for Publishing Linked Data.* Opgehaald van https://www.w3.org/TR/ld-bp/

W3C. (sd). *SKOS.* Opgehaald van https://www.w3.org/TR/skos-reference/.

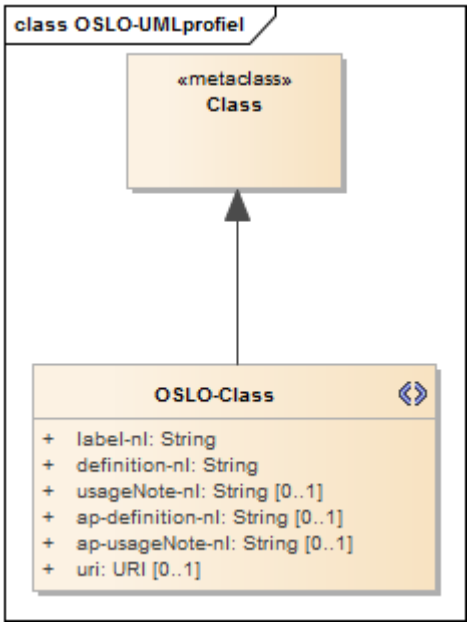
ANNEX A: UML-PROFIEL

Een UML-profielen laat toe om een metamodel aan te passen aan het eigen domein of platform. Dit gebeurt door het metamodel uit te breiden met nieuwe elementen (zgn stereotypes) of door kenmerken aan bestaande elementen toe te voegen of bijkomende constraints op te leggen.

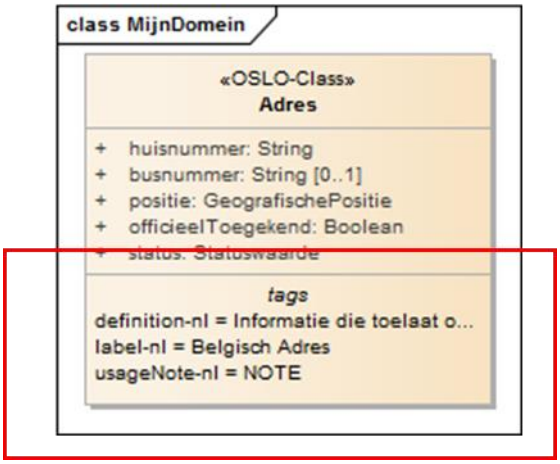
Het OSLO UML-profiel is een aanpassing vh UML-metamodel omdat we UML als modelleringstaal gebruiken (zie §3.1.8). De aanpassing beperkt zich tot het toevoegen van nieuwe kenmerken, meer bepaald de door de regels voorgeschreven metadata (zie §3.2.19):

- Label
- Definitie
- Gebruiksnota
- Uri

Hiertoe werd van elk standaard element (de zgn metaklassen) een stereotype afgeleid met deze kenmerken, bv van metaklasse Class leidden we volgend stereotype OSLO-Class af:



Bij instantiatie van OSLO-Class materialiseren de toegevoegde attributen zich als valued tags, ttz “data over data” of metadata. Voor bovenstaand vb ziet dit er zo uit:



Merk op dat overbodige tags werden verwijderd, zie §3.2.19 voor informatie wanneer welke tags nodig zijn.

Volgende stereotypes zijn momenteel voorzien ih OSLO UML-profiel:

Stereotype	**Metaklasse**
OSLO-Package	Package
OSLO-Class	Class
OSLO-Datatype	Datatype
OSLO-Enumeration	Enumeration
OSLO-PrimitiveType	PrimitiveType
OSLO-AssociationClass	AssociationClass
OSLO-Attribute	Attribute
OSLO-AssociationEnd	AssociationEnd
OSLO-Association	Association
OSLO-Aggregation	Aggregation
OSLO-Composition	Composition
OSLO-Generalization	Generalization

Waarvan momenteel enkel OSLO-Package, OSLO-Enumeration en OSLO-Generalization geen toegevoegde kenmerken hebben.

Het profiel bevat geen domeinspecifieke of platformspecifieke stereotypes. Het enige “domein” dat het profiel ondersteunt is dat vh semantisch web, dit door de mogelijkheid om de semantiek te documenteren dmv metadata zoals label, definitie, uri etc te ondersteunen.

Een werkelijk domeinspecifiek stereotype zoals bv <<Adres>> is niet aan te bevelen omdat de semantiek van bv een concept als Adres dan verschuift van het VOC/AP naar het profiel. Tags als label, definitie, uri etc hebben geen zin, deze info gaat bij instantiatie verloren.

Ook platformspecifieke stereotypes ontbreken ih profiel aangezien die vh conceptueel model of applicatiemodel een implementatiemodel zouden maken (zie §2.4).

Het volledige profieldiagram ziet er als volgt uit: