# Metamodel
# voor informatiemodellen




versie 1.0
14 juni 2017

FINAL versie

*[Toelichting FINAL versie]*

Deze versie is voorgelegd voor officiele besluitvorming en heet daarom versie 1.0 FINAL. Dit document wordt gezien als geheel stabiel.

Mogelijk dat er nog een versie uitkomtwaarin nog wat meer gekeken is naar opmaak t.b.v. printvriendelijkheid, leesbaarheid en spelling. Dit betreft uiteraard géén inhoudelijke wijzigingen.

Aan dit metamodel zal, naar behoefte, verder gewerkt worden. Bijvoorbeeld naar aanleiding van nieuwe inzichten of nieuwe wensen. Ook zullen er tijdens het gebruik vragen ontstaan en verhelderingen nodig zijn, welke te zijner tijd ook opgenomen kunnen worden. Deze worden niet meer meegenomen in deze versie 1.0. maar kunnen dus prima opgenomen worden in een volgende versie.

**Colofon**

Aan de totstandkoming van dit metamodel hebben de volgende inhoudelijke experts op het gebied van informatiemodellering meegewerkt:

| Naam | Organisatie | Rol en achtergrond |
|---|---|---|
| Ir. Ellen Debats | KING | Beheerder KKG. KKG kerngroep. Expert gegevenstandaarden. Gemma standaarden, RSGB/RGBZ/imZTC. |
| Ir. Lennart van Bergen | Kadaster | Beheerder KKG. KKG kerngroep. Expert informatiemodellering. Modellenbureau Kadaster, IMKAD, IMBAG. Vanuit DSO betrokken bij LVBB (PR30) en de catalogus (PR06). |
| Ir. Paul Janssen | Geonovum | Beheerder KKG. KKG kerngroep. Expert GEO standaarden. NEN3610. IMRO, IMKL. Vanuit DSO betrokken bij informatiemodel standaarden Omgevingswet (PR04). |
| Ir. Arjan C. Kloosterboer | KING | KKG kerngroep. Gegevensarchitect GEMMA. |
| Ing. Peter Lentjes | Kadaster | KKG kerngroep. Modellenbureau Kadaster, IMBRT, IMGEO (BGT) |
| Drs. Wilko Quak | TU Delft | KKG kerngroep. Expert GEO standaarden. NEN3610. TU Delft. |
| Drs. Linda van de Brink | Geonovum | KKG kerngroep. Expert GEO standaarden. NEN3610, BRO, IMGEO (BGT) |
| Drs. Thies Mesdag | Kadaster | Reviewer. Modellenbureau Kadaster, IMKAD. |
| Drs. Arjan Loeffen | Armatiek BV | Reviewer. Expert model driven design. Implementatie in tooling. |
| Drs. Jos Warmer | OpenModeling | Reviewer. Expert UML, model driven design, UML. |

**Voorwoord**

Informatie is een belangrijke motor voor het functioneren van de overheid in Nederland. Steeds meer wordt er samengewerkt en uitgewisseld tussen verschillende overheidslagen en instanties. Daarom is het van groot belang dat we hetzelfde verstaan onder de gegevens die we gebruiken en gemeenschappelijke afspraken maken over de wijze van beschrijven van gegevens en de manier waarop we deze uitwisselen.

Bij de huidige ontwikkeling van het Digitaal Stelsel Omgevingswet (DSO) komt duidelijk naar voren dat veel informatiebronnen bij elkaar komen in de zogenaamde informatiehuizen. Daarbij komen ook de wereld van de geografische gegevens en de wereld van de meer administratieve registraties bij elkaar. Dit geeft niet alleen extra mogelijkheden om informatie slim te combineren maar maakt ook een aantal verschillende uitgangspunten zichtbaar in de werkwijze van Kadaster, KING en Geonovum als het gaat om het modelleren van informatie.

Met het voorliggende KKG-metamodel komen we tot een gemeenschappelijk vertrekpunt voor het opstellen van informatiemodellen. Het voorziet enerzijds in duidelijke afspraken over het vastleggen van gegevensspecificaties en biedt anderzijds ruimte aan de verschillende niveaus van modellering. Bijzonder aan het KKG model is dat we afspraken maken die over meerdere bestuurslagen heen gaan.

Wij zijn er van overtuigd dat het KKG-metamodel gaat leiden tot vergelijkbare modellen voor delen van de overheidsinformatievoorziening, zoals bijvoorbeeld de informatiehuizen in het DSO, en houvast geeft bij het opstellen van informatiemodellen ongeacht het beschouwingsniveau dat van toepassing is. Vanuit Kadaster, KING en Geonovum dragen wij op deze wijze graag bij aan betere afspraken over het beschikbaar stellen en uitwisselen van informatie.

Arjen Santema (Kadaster), Theo Peters (KING) en Marcel Reuvers (Geonovum)
April 2017

**Inhoud**

# 1. Inleiding

Voor u ligt het metamodel voor het beschrijven van informatiemodellen. Aanleiding was oorspronkelijk het adviesrapport 'Rapportage harmonisatie StUF en NEN 3610' (KING/Geonovum, 2010) waarin één van de aanbevelingen was om één modelleertaal, UML, te gebruiken voor informatiemodellen. Het belang van afstemming tussen het Stelsel van Basisregistraties en de NEN3610-informatiemodellen en recente ontwikkelingen zoals onder andere het Digitaal Stelsel Omgevingswet, maken de noodzaak om te komen tot één modelleertaal urgent.

Afspreken dat we UML als modelleertaal gebruiken is hierbij niet voldoende. Een metamodel is nodig. Dit is een verzameling van de bouwstenen c.q. de modelelementen die gebruikt mogen worden om een informatiemodel mee op te stellen. Het is de modelleertaal waarin een informatiemodel is uitgedrukt. Toepassing van het metamodel leidt tot eenduidige interpretatie van de informatiemodellen en bevordert de uitwisselbaarheid van deze modellen binnen de eigen organisatie en – vooral - daarbuiten.

Dit document is opgesteld door KING, Kadaster en Geonovum. De kennis en kunde die deze organisaties hebben opgedaan in de jarenlange praktijk van het opstellen van informatiemodellen, is hierin samengebracht.

## 1.1 Toepassingsgebied

Het metamodel biedt de modelleringstaal waarmee een informatiemodel gemaakt, gelezen en begrepen kan worden. Het doel hiervan is:

- de leesbaarheid en eenduidigheid van informatiemodellen te vergroten;
- informatiemodellen op zowel conceptueel als op logisch niveau te kunnen opstellen (zie par. 1.4);
- informatiemodellen goed aan elkaar te kunnen koppelen op conceptueel en logisch niveau;
- tooling te kunnen ontwikkelen en (her)gebruiken voor resp. door alle partijen die kiezen voor dit metamodel;
- kennis uit verschillende organisaties te bundelen;
- en in het verlengde hiervan, om op een meer geautomatiseerde werkwijze sneller en beter voorspelbaar uitwisselingsstandaarden op kunnen stellen. en
- om interoperabiliteit tussen registraties te bevorderen.

Voor informatiemodellen die op basis van dit metamodel zijn beschreven geldt:

- ze zijn eenduidig te interpreteren en goed te vergelijken;
- er kan documentatie mee opgesteld of gegenereerd worden welke geschikt is voor publicatie;
- ze kunnen als basis gebruikt worden voor (bij voorkeur model-driven generatie van) afgeleide modellen en producten voor een specifiek toepassingsgebied/domein zoals bijvoorbeeld NEN3610 of het gemeentelijke domein;
- ze kunnen als basis gebruikt worden voor (bij voorkeur model-driven generatie van) afgeleide modellen voor specifieke services en informatieproducten (implementatieschema's, registers, validatieservices e.d.).

## 1.2 Doelgroep

Dit document is primair bestemd voor informatiearchitecten die deze informatiemodellen maken; informatieanalisten die willen weten wat de betekenis en definitie van informatieobjecten is, en mensen die model-driven verder werken op basis van het informatiemodel en er implementaties van maken. Kennis van informatiemodellering is een vereiste. Enige kennis van UML is een pré maar niet noodzakelijk. Dit metamodel richt zich in het bijzonder op de informatievoorziening binnen het overheidsdomein, al is het ook in bredere context inzetbaar.

## 1.3 Leeswijzer

Het metamodel beschrijven we in drie hoofdstukken en een bijlage.

Lees dit hoofdstuk (1 – Inleiding) verder voor inzicht in wat we onder een informatiemodel en onder een metamodel verstaan, hoe deze de modellen zich verhouden tot UML en de vier lagen metamodel architectuur van de Object Management Group (OMG), en welke standaarden worden toegepast.

Hoofstuk 2 – Metamodel bevat de beschrijving van alle bouwstenen c.q. de modelelementen van het metamodel, in de vorm van definities en specificaties. Ook beschrijft dit hoofdstuk hoe het metamodel zich verhoudt tot het UML metamodel, welke uitbreidingen c.q. verbijzonderingen van het UML metamodel zijn aangebracht. De betekenis en toelichting van de modelelementen van het metamodel vormt het materiaal waarmee een uitputtende modelspecificatie kan worden opgesteld.

In hoofdstuk 3 – 'Overige afspraken en regels' gaan we in detail in op een aantal aspecten. Het is een uitgebreidere toelichting, in aanvulling op hoofdstuk 2, bestaande uit nadere afspraken, regels, richtlijnen en aanbevelingen bij het toepassen van het metamodel.

Bijlage 3 verschaft een overzicht van alle bouwstenen en metadata-elementen en het al dan niet van toepassing zijn daarvan in een conceptueel dan wel een logisch informatiemodel.

## 1.4 Wat is een informatiemodel

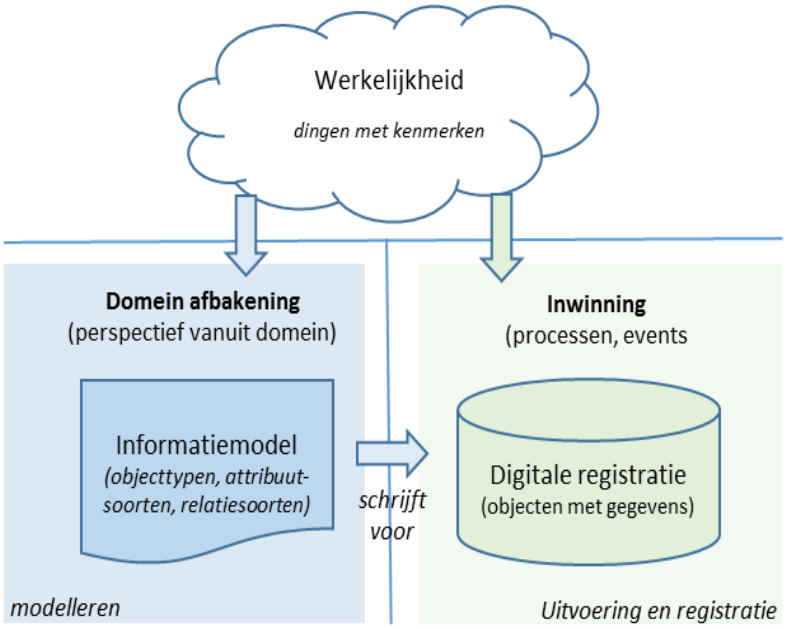
Een informatiemodel beschrijft de structuur, semantiek en de eigenschappen van informatie over dingen in de werkelijkheid. Met semantiek wordt de betekenis en definitie van de informatie over 'het ding' bedoeld, onafhankelijk van een mogelijke implementatie of toepassingsomgeving. Er worden dus geen regels toegepast die gerelateerd zijn aan de manier waarop de gegevens ingewonnen, opgeslagen, beheerd en uitgewisseld worden. Die beschrijving heeft de vorm van een model dat een gestructureerde weergave is van die werkelijkheid. Een dergelijk model is noodzakelijk om deze informatie te kunnen beheren en gebruiken (door mensen en machines) bij het communiceren over deze werkelijkheid, in registraties[1] of anderszins, zoals het specificeren van de tussen registraties uit te wisselen gegevens of van de te bevragen informatie uit een registratie.

Het beschrijven vindt plaats door de informatie te modelleren naar objecttypen en de kenmerken daarvan naar attribuutsoorten van die objecttypen en relaties tussen die objecttypen. Alleen dingen en kenmerken die relevant zijn voor een bepaald domein worden in het informatiemodel beschreven, zoals gebouwen binnen het domein Basisregistratie Topografie en personen binnen het domein Basisregistratie Personen. Een domein kan van alles zijn maar in het kader van dit metamodel gaat het om (beleids)sectoren die omwille van bestuurlijke en beheersmatige redenen geïdentificeerd en georganiseerd zijn. Voorbeelden: ruimtelijke ordening, grootschalige topografie, kadastrale informatie of gemeentelijk domein.

Objecttypen in een informatiemodel representeren dingen in de werkelijkheid. De volgende tekst beschrijft hoe dingen in de werkelijkheid zich verhouden tot het informatiemodel. We visualiseren dat in onderstaande figuur voor de situatie dat er een, van het informatiemodel afgeleide, registratie is.

[1] De opname in een registratie kent vaak een inwinningsproces, om gegevenswaarden over de feitelijke dingen in de werkelijkheid conform het informatiemodel in de registratie op te nemen. Dit is een belangrijk proces, maar valt buiten scope van het informatiemodel.

Jan en Katrien zijn bijvoorbeeld ‘dingen in de werkelijkheid’. Zij hebben bepaalde kenmerken, zoals een naam en een geboortedatum. In een informatiemodel komen Jan en Piet niet voor. Ook hun gegevens, zoals het feit dat 10-10-1970 de geboortedatum van Jan is, komt niet voor.




In de context van het informatiemodel worden Jan en Katrien gezien als *objecten* binnen een domein. In het informatiemodel is het objecttype Persoon gedefinieerd en Jan en Piet zijn dus objecten van het objecttype Persoon.

De *kenmerken* zoals de naam en geboortedatum maar bijvoorbeeld ook identificatie en registratietijdstip worden gezien als attributen van dit objecttype. We noemen een dergelijk kenmerk een attribuutsoort. Sommige kenmerken, zoals het gegeven dat Jan getrouwd is met Katrien, modelleren we door middel van een relatiesoort tussen objecttypen, in dit geval van Persoon met zichzelf.

In de van het informatiemodel afgeleide registratie kunnen vervolgens de objecten Jan en Katrien en de gegevens ervan, zoals de geboortedatum 10-10-1970, worden vastgelegd.

Als een andere registratie op haar eigen manier tegen dezelfde ‘Jan uit de werkelijkheid’ aankijkt, dan is ook in die registratie een (eigen, apart) object voor Jan aanwezig en Jan kan in dit (eigen, apart) informatiemodel anders gemodelleerd zijn. Beide objecten Jan representeren natuurlijk dezelfde ‘Jan uit de werkelijkheid’, vanuit het perspectief van het eigen domein bekeken.

## 1.5 Typen informatiemodellen

Zoals hiervoor uiteengezet beschrijft een informatiemodel de werkelijkheid. In de praktijk blijken hier niveaus in te bestaan, variërend van een zo getrouw mogelijke beschrijving van die werkelijkheid tot een specificatie van de wijze van vastlegging van die werkelijkheid in een database of uitwisselformaat. Veelal worden vier niveaus onderscheiden[2]:

1. ***Model van begrippen***
   Beschrijft de werkelijkheid binnen het beschouwde domein (de ‘universe of discourse’) d.m.v. de daarin gehanteerde begrippen en hun relaties tot elkaar. Doel is dat de actoren daarbinnen elkaar begrijpen en één taal spreken. Een model van begrippen wordt opgesteld voor gebruik door mensen, met name ‘de business’. De begrippen worden beschreven in een formele taal, een vocabulaire. Een vocabulaire is geen informatiemodel. Begrippen kunnen in meerdere informatiemodellen gebruikt worden.

2. ***Conceptueel informatiemodel***
   Modellering van de werkelijkheid binnen het beschouwde domein, v.w.b. informatie daarvan, onafhankelijk van ontwerp van en implementatie in systemen. Het geeft een zo getrouw mogelijke beschrijving van die werkelijkheid en is in natuurlijke taal geformuleerd. Een dergelijk model definieert

[2] Ontleend aan: Model Driven Architecture (MDA) Guide; Object Management Group, rev. 2.0, 1-6-2014

het ‘wat’: welke ‘concepten’ (‘dingen’) worden onderscheiden (in de beschouwde werkelijkheid), wat betekenen zij, hoe verhouden ze zich tot elkaar en welke informatie (eigenschappen) is daarvan relevant. Het dient als taal waarmee domeinexperts kunnen communiceren met informatie-analisten en verschaft een eenduidige interpretatie van die werkelijkheid ten behoeve van deze communicatie. Een conceptueel informatiemodel wordt dan ook opgesteld voor gebruik door mensen, zodat ‘de business’ en de ICT-specialisten elkaar gaan begrijpen.

3. ***Logisch informatie- of gegevensmodel***
   Beschrijft hoe de, in het conceptuele model onderscheiden, concepten gebruikt worden bij de interactie tussen systemen en hun gebruikers en tussen systemen onderling. Anders gezegd, een model van de representatie van informatie over de werkelijkheid in digitale registraties en in de uitwisseling daartussen. Het gaat hierbij, in tegenstelling tot een conceptueel model, dus veel meer om het ‘hoe’. Het slaat de brug tussen werkelijkheid en systemen maar beschrijft nog niet de implementatie in die systemen. Een dergelijk model wordt in een formele taal beschreven en wordt waar mogelijk gegenereerd vanuit het conceptueel model. Het logisch model wordt opgesteld voor ICT-interoperabiliteit, voor gebruik door met name de ontwerpers, bouwers en beheerders van ICT-voorzieningen.

4. **Fysiek of technisch gegevens- of datamodel**
   Specificeert de structuur en eigenschappen van de technologie waarin de informatie wordt vastgelegd of uitgewisseld. Dit is sterk afhankelijk van de gebruikte opslagtechnologie zoals een specifieke database of de servicetechnologie zoals XML, GML, SOAP, REST, (Geo)JSON, LinkedData e.d. Het kan tevens informatie bevatten over de manier waarop berichten ‘verpakt’ worden, het (internet)protocol en de logistiek van het berichtenverkeer. De technische specificaties worden over het algemeen zoveel als mogelijk gegenereerd uit het logisch informatiemodel. Deze specificaties worden opgesteld voor ‘machines’, te gebruiken door software-ontwikkelaars.

Het voorliggende metamodel kan toegepast worden op twee niveaus, niveau 2 en niveau 3: t.b.v. een zuiver conceptueel informatiemodel (2) en t.b.v. een logisch informatiemodel (3). Het moge duidelijk zijn dat het altijd het één of het ander is. Een combinatie van beide in één model leidt tot verwarring. Voor eenzelfde domein verschilt de structuur van het informatiemodel naar gelang het type en bevat het logisch informatiemodel meer, vooral formele, specificaties dan een conceptueel model. Het is voor de hand liggend maar niet persé noodzakelijk om voor een domein eerst een conceptueel en daarna een logisch informatiemodel op te stellen. Een organisatie kan er voor kiezen om alleen een logisch informatiemodel op te stellen. Een begrippenmodel is in dat geval dringend aanbevolen. Bijlage 3 verschaft een overzicht van de metadata-constructen en -elementen die per type model van toepassing zijn.

Het is van belang om voorafgaand aan het opstellen van een informatiemodel expliciet te bepalen welk van beide typen beoogd is en de modellering conform het gekozen type te doen plaatsvinden. In de beschrijving van het informatiemodel moet vermeld worden om welk van beide typen het gaat. Aan te bevelen is om eerst een conceptueel model op te stellen en dit vervolgens uit te werken naar een logisch model.

## 1.6 Wat is het metamodel voor informatiemodellen

Een metamodel is een model van een model. Het definieert een verzameling van modelleerconstructies in de vorm van bouwstenen oftewel modelelementen, met bijbehorende betekenis en bijbehorende afspraken omtrent hoe deze toe te passen. Een informatiemodel kan vervolgens hiermee gemaakt worden. Het metamodel is daarmee de modelleertaal waarin een informatiemodel is uitgedrukt. Deze metataal beschrijft als het ware de grammatica en de syntax van de modelleertaal.

Vaak zie je dat het metamodel niet expliciet beschreven is en dat het metamodel een onderdeel van de domeinkennis is geworden. Bij domeinoverstijgende harmonisatie wordt het dan moeilijk om modellen met elkaar te vergelijken en op basis daarvan gegevens uit te wisselen. Beschrijving van het metamodel is daarom een randvoorwaarde indien er sprake is van een stelsel van samenhangende informatiemodellen. Anders gezegd, met (alleen) de in het metamodel opgenomen set van modelleerconstructies worden informatiemodellen gemaakt. Door het schrijven van modelleertalen (zoals UML) in een metataal (zoals MOF) wordt gegarandeerd dat alle toepassingen van die talen op een standaard manier zijn opgebouwd en daardoor alom te begrijpen zijn. De metataal beschrijft als het ware de grammatica van de modelleertaal. Het metamodel in dit document is gebaseerd op UML.

## 1.7 UML

Voor zowel het metamodel als informatiemodellen wordt uitgegaan van UML. Registraties en afnemers hiervan kunnen deze gebruiken voor de inrichting van hun situatiespecifieke gegevenshuishouding. Belangrijk is dat de lezer eerst begrijpt wat we onder een informatiemodel en een metamodel verstaan en verder is het van belang de modellen in de juiste context te plaatsen. Dit laatste doen we aan de hand van de vier lagen metamodel architectuur van de Object Management Group (OMG). In deze paragaaf gaan we op deze concepten in.

**Vier lagen metamodel architectuur OMG**

Voor de specificatie van het metamodel is gebruik gemaakt van dezelfde formele taal waarin de informatiemodellen zijn beschreven, namelijk UML. Het metamodel van deze informatiemodellen is een uitbreiding op het basale UML-metamodel.

Het basale UML-metamodel is een metamodel dat onderdeel uitmaakt van de vier lagen metamodel architectuur van OMG namelijk M0, M1, M2 en M3. Daarbij is elke laag een instantie van de laag daarboven (met uitzondering van de 1e laag) en maakt de laag gebruik van een in de naast hoger gelegen laag gespecificeerde uitdrukkingsmogelijkheden teneinde een specificatie in een andere context te vormen. De toplaag is de metamodellaag oftewel M3 laag en definieert de basisconstructies, m.a.w. de taal waarin de onderliggende laag is uitgedrukt. Metamodel Meta Object Facility (MOF) is een voorbeeld van deze laag. MOF is de basislaag voor de UML laag. De metamodel laag (M2) is een instantie van de M3 laag. Op deze laag bevindt zich onder meer het metamodel UML. M.a.w. UML is een instantie van MOF. Deze laag is taaltechnisch rijker dan de M3 laag. De M2 laag definieert de semantiek en syntax van de modelconstructies in de M1 laag. De M1 laag is de laag waarop zich het informatiemodel bevindt om een bedrijfscontext modelmatig te beschrijven. Deze M1 laag is een instantie van de M2 laag. Tenslotte is er nog de M0 laag waarop zich de objecten en data bevinden, de instanties van de M1 modelconstructies die een representatie van de concrete werkelijkheid op een specifiek tijdsmoment vormen.

| Metaniveau | Omschrijving | Elementen |
|---|---|---|
| M3 | MOF, verzameling van constructies voor definiëren van metamodellen | MOF klasse, MOF attribuut, MOF Associatie, MOF operatie, etc. |
| M2 | Metamodellen (UML, CWM, etc.), bestaande uit instanties van MOF constructies | UML klasse, UML associatie, UML attribuut, etc. |
| M1 | Modellen, bestaande uit instanties van M2 metamodel constructies | Klasse “Order”, klasse “Klant”, attribuut “naam” etc |
| M0 | Objecten en data, de instanties van M1 model constructies | Order 43123, Artikel 8RB31, etc. |

**Tabel 1 Vier lagen metamodel OMG**

De informatiemodellen waarover we het hier in dit document hebben bevinden zich op de M1-laag.

Dit metamodel is een uitbreiding op het UML metamodel (M2). Het UML metamodel is daarbij uitgebreid met speciale elementen, die geen onderdeel uitmaken van het basale UML-metamodel (M2). Deze nieuwe elementen zijn noodzakelijk voor het definiëren van de semantiek en syntax van de modelconstructies zoals we die in onze informatiemodellen hanteren.

Het UML metamodel (M2) is een 'read only' model. Dat wil zeggen dat we geen bestaande metaclass mogen aanpassen en we dus geen nieuw basis metaclass voor een bestaande UML metaclass mogen specificeren. Maar via Profiles (van de InfrastructureLibrary) kunnen bestaande metaclasses uitgebreid worden zonder dat er nieuwe metaclasses gedefinieerd hoeven te worden en dus zonder aanpassing van het basale UML-metamodel (M2). De extensiemechanismen hiervoor zijn stereotypes, tagged values en constraints.

## 1.8 Een eigen extensie op het metamodel

Indien er extra metamodelconstructies nodig zijn voor een informatiemodel, dan kan dit metamodel uitgebreid worden met een aanvulling oftewel extensie (in de vorm van een extra bijlage) die door de betreffende organisatie toegevoegd wordt aan het onderhavige document.

De spelregel bij een extensie is dat deze geen onderwerpen *vervangt* die in dit metamodel beschreven zijn, maar alleen echte uitbreidingen behelst. Indien meerdere organisaties hierin geïnteresseerd zijn, kan zo'n extensie ook toegevoegd worden aan dit metamodel. Neem dan contact op met de beheerders (zie voorwoord).

Het is ook mogelijk om in de extensie aan te geven welke elementen uit dit metamodel *niet* ingezet (mogen) worden in uw informatiemodellen. Denk hierbij bijvoorbeeld aan een bepaald modelelement. Of aan bepaalde metadata aspecten die niet ingewonnen worden in uw informatiemodellen en daarom buiten scope worden geplaatst (ongeacht of deze optioneel of verplicht zijn).

Voor meer informatie over een specifieke extensie kan contact opgenomen worden met de beheerder van deze extensie.

Nota bene: een metamodel extensie is expliciet niet bedoeld voor aanvullende constructies die alleen spelen op het niveau van implementatie, of op het niveau van afgeleide modellen t.b.v. specifieke koppelvlakken en interfaces. Deze vallen buiten scope van dit metamodel en ook buiten scope van extensies hierop. Wel is het mogelijk en toegestaan om het metamodel, of delen ervan, hiervoor te gebruiken.

## 1.9 Alternatieven

In dit metamodel is op één punt sprake van een keuze tussen twee alternatieven, waarvan de modelleur van een informatiemodel één van beide alternatieven kiest. Welke je kiest geef je aan bij je eigen informatiemodel, in je eigen extensie (zoals bedoeld in de vorige paragraaf).

Dit betreft: Relatiesoort en relatierol, beide te gebruiken, maar welke is verplicht/leidend (paragraaf 2.3.2.1 en 2.3.2.2).

Indien gewenst kunt u hier vragen over stellen aan de beheerders van dit metamodel voordat u een keuze maakt.

## 1.10 Beheer

Het beheer van dit metamodel vindt plaats als samenwerking tussen KING, het Kadaster modellenbureau en Geonovum. Voor vragen, suggesties of opmerkingen kunt u contact opnemen met:

Ellen Debats: ellen.debats@kinggemeenten.nl en/of
Lennart van Bergen: lennart.vanBergen@kadaster.nl en/of
Paul Janssen: p.janssen@geonovum.nl

## 1.11 Normreferenties

| # | Naam | **Referentie** |
|---|---|---|
| 1. | Unified Modeling Language (UML) | http://uml.org |
| 2. | OMG Unified Modeling Language TM versie 2.5 | http://www.omg.org/spec/UML/2.5 |
| 3. | Stelselcatalogus | http://www.stelselcatalogus.nl/over-de-stelselcatalogus/metadata/ |
| 4. | GAB | http://www.noraonline.nl/images/noraonline/c/c3/GAB_mogelijk_onvolledige_datum_1.0.pdf |
| 5. | Handreiking gegevensbeschrijving (NORA) | http://noraonline.nl/wiki/Gegevensbeschrijvingen/Handreiking |
| 6. | ISO 11404 | NEN-ISO/IEC 11404:2008 Information technology – General Purpose Datatypes (GPD) |
| 7. | ISO 8601 | https://en.wikipedia.org/wiki/ISO_8601 |
| 8. | Formeel patroon (Reguliere Expressies) | Zoals beschreven in Perl 5: http://perldoc.perl.org/perlre.html |
| 9. | OCL | http://www.omg.org/spec/OCL/2.4/ |
| 10. | NEN 3610/A1:2016 Basismodel Geo-informatie. | NEN 3610:2011/A1:2016 nl. Basismodel geo-informatie - Termen, definities, relaties en algemene regels voor de uitwisseling van informatie over aan de aarde gerelateerde ruimtelijke objecten |

De Stelselcatalogus, het GAB en de Handreiking gegevensbeschrijving raken elkaar op een aantal vlakken maar er kunnen op deze raakvlaken verschillen zijn in de gemaakte afspraken. Voor het metamodel hanteren we daarom de volgende spelregel: de Stelselcatalogus is zoveel als mogelijk leidend, vervolgens het GAB en als laatste de handreiking.

# 2. Metamodel

Dit hoofdstuk beschrijft het metamodel in diagramvorm en in tekst. De eerste paragraaf bevat diagrammen, in UML; elk biedt een eigen view op een gedeelte van het model. Het geheel van diagrammen, in samenhang, is opgenomen in bijlage 3.

Uitgangspunten voor het metamodel zijn:

- UML 2.5 vormt de basis voor de conceptuele beschrijving.
- Gebruik te maken van de bestaande UML- modelelementen conform UML van OMG. OMG noemt dit een UML metaclass. Een voorbeeld hiervan is UML-Class.
- Daar waar (semantisch) nodig extensiemechanismen toe te passen met behoud van de betekenis van de UML-metaclasses. Er ontstaat dan een KKG metaclass. Hoe deze zich verhouden tot UML is weergegeven in de bijlage.
- Modelelementen hebben één stereotype. Daarnaast hebben twee verschillende stereotypen nooit dezelfde betekenis. Stereotypes worden toegepast als er een verbijzondering van een UML constructie nodig is met behoud van de betekenis van de UML-metaclass.
- Uniforme toepassing van het metamodel in informatiemodellen. Anders gezegd, uitbreiden mag, afwijken niet, maak voor hetzelfde doel geen alternatieve constructies.
- Datatypen zijn onderdeel van het metamodel en beschrijven de structuur van de data, maar niet de semantiek/betekenis. De aanbeveling is dan ook om eerst een informatiemodel te maken zonder datatypen. De regel is dat als alle datatypen uit het model worden weggelaten, er geen semantische betekenis verloren mag gaan.
- Toolonafhankelijke beschrijving van het metamodel. Omdat KING, Kadaster en Geonovum en veel andere organisaties Sparx EA gebruiken is er aanvullend aangegeven hoe het metamodel in Enterprise Architect toegepast wordt. Hierdoor borgen we deze relatie.

## 2.1 Structuur metamodel

Deze paragraaf bevat een overzicht van het metamodel en geeft alle modelelementen weer in diagram vorm. De beschrijving van de modelelementen in tekst vorm staan in de volgende paragraaf.

De modelelementen zijn verdeeld over een diagrammen, die elk een eigen view op een deel van het metamodel tonen. Elk view toont een aantal van de modelelementen, inclusief hun onderlinge samenhang. Alle views samen vormen het metamodel als geheel:

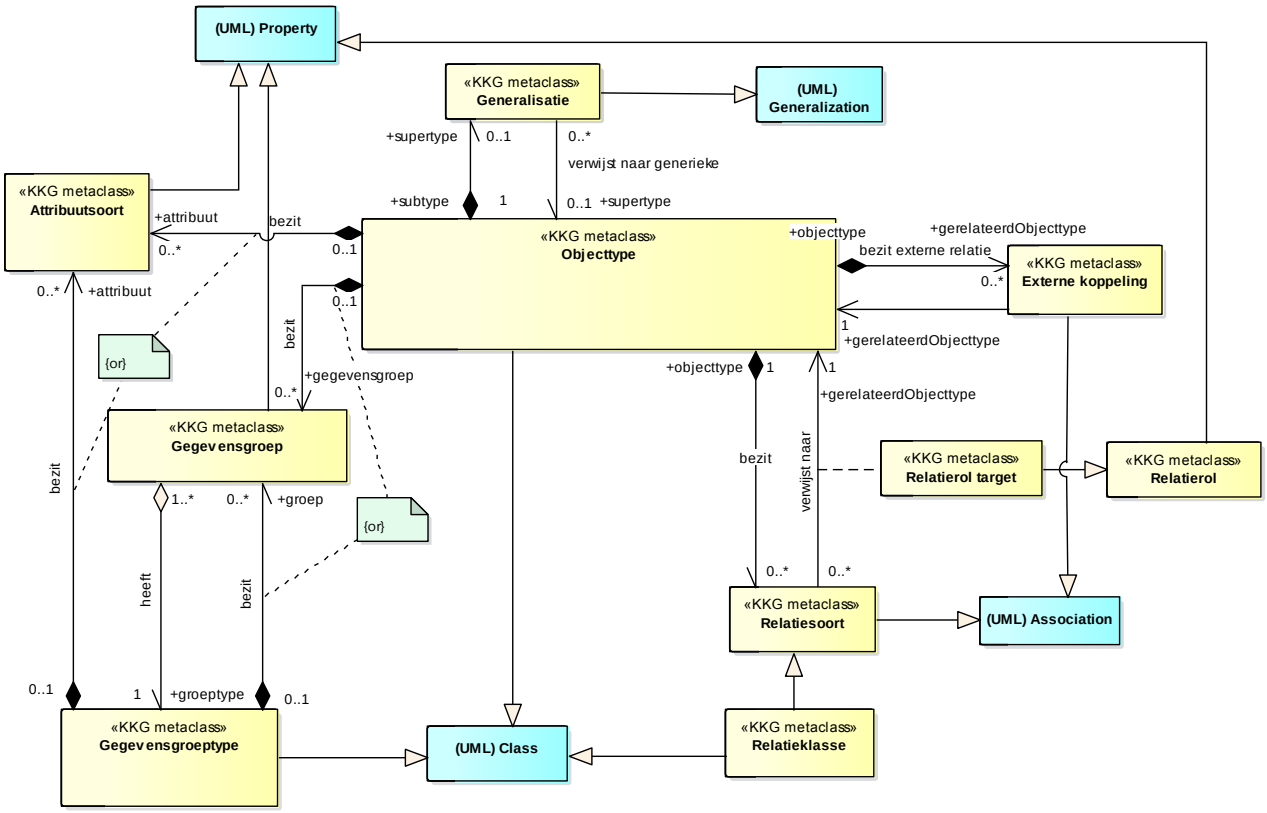
- KERN, met de belangrijkste modelelementen in onderlinge samenhang.
- DATATYPEN, met de in het model te onderkennen soorten datatypen.
- OVERIGE modelelementen, die niet altijd aan de orde zijn.

Elk modelelement heeft een KKG metaclass. Deze wordt in UML in een informatiemodel gemodelleerd als een Metaclass van UML 2.5 en een bijbehorende stereotype. Bijvoorbeeld: de KKG metaclass Objecttype wordt gemodelleerd als een UML-Class met stereotype «Objecttype». In Sparx EA wordt dit gemodelleerd met een Class. Niet alle KKG metaclasses hebben een stereotype (nodig). In de kolom staat dan '-'.

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Objecttype | «Objecttype» | (UML) Class | Class |

In de diagrammen zijn de UML metaclasses conform UML 2.5 aangeduid als UML metaclass. Deze in opgenomen in het diagram als 'blauw gekleurde' metaclasses.

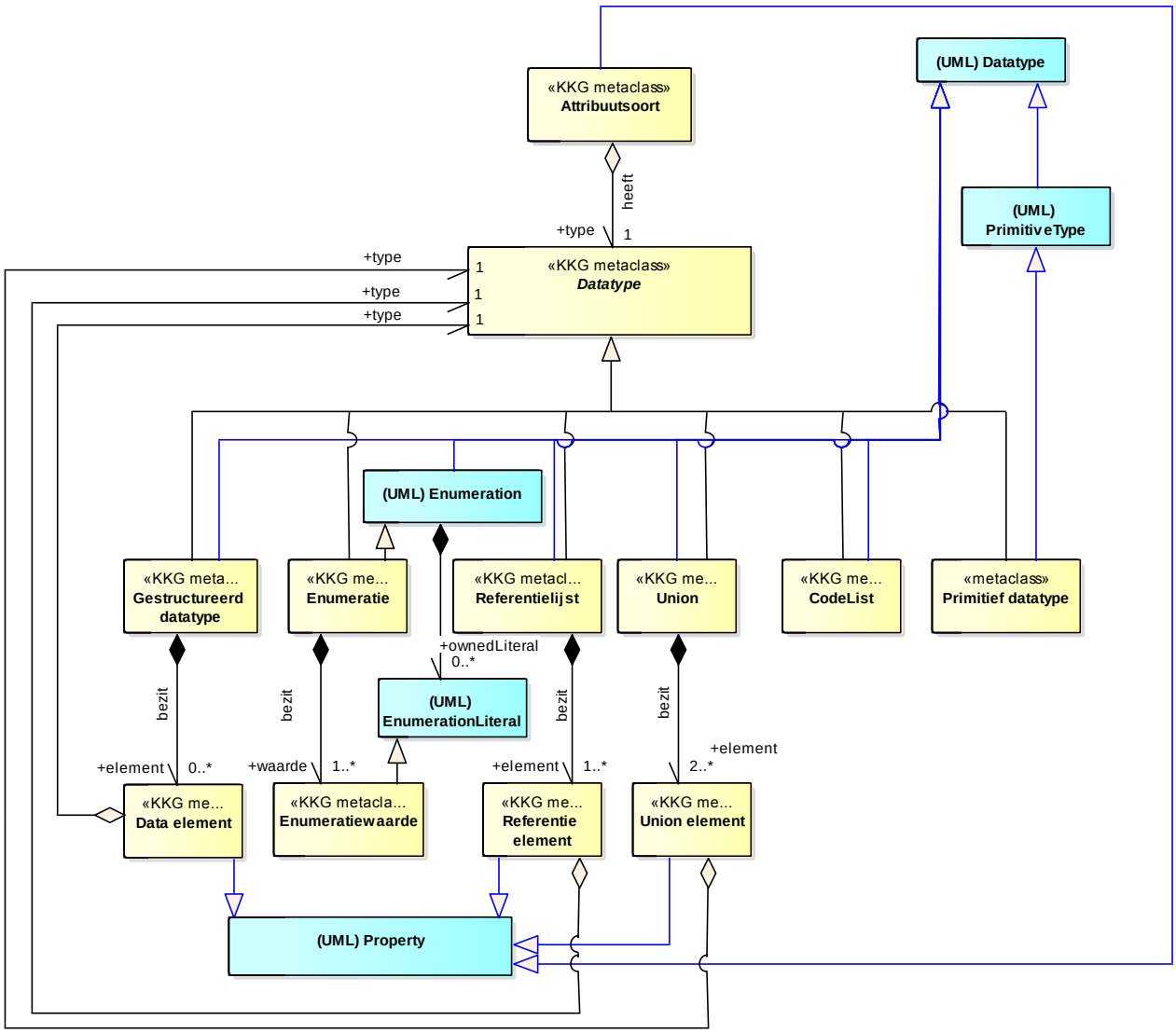
### 2.1.1 Kern




**View 1: Kern metamodel**

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Objecttype | «Objecttype» | (UML) Class | Class |
| Attribuutsoort | «Attribuutsoort» | (UML) Property | Attribute |
| Gegevensgroep | «Gegevensgroep» | (UML) Association | Association |
| Gegevensgroeptype | «Gegevensgroeptype» | (UML) Class | Class |
| Generalisatie | «Generalisatie» | (UML) Generalization | Generalization |
| Relatiesoort | «Relatiesoort» | (UML) Association | Association |
| Relatieklasse | «Relatieklasse» | (UML) Association én (UML) Class | Associationclass |

### 2.1.2 Datatypen

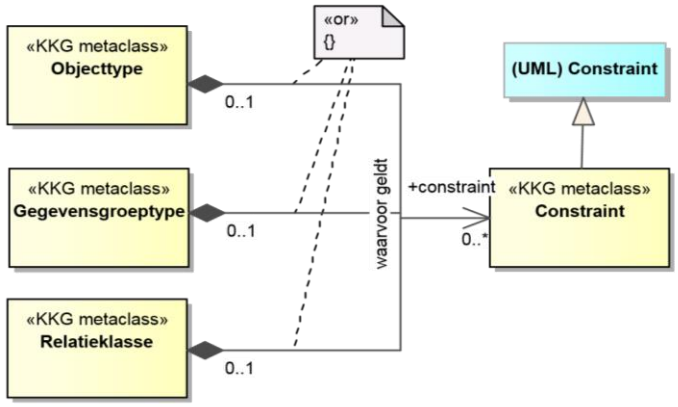



**View 2: Datatypen**

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Primitief datatype | «Primitief datatype» | (UML) Primitive Type | Datatype |
| Gestructureerd datatype | «Gestructuurd datatype» | (UML) Datatype | Datatype |
| Data element | «Data element» | (UML) Property | Attribute |
| Union | «Union» | (UML) Datatype | Datatype |
| Union element | «Union element» | (UML) Property | Attribute |
| Enumeratie | - | (UML) Enumeration | Enumeration |
| Enumeratiewaarde | - | (UML) EnumerationLiteral | EnumerationLiteral |
| Referentielijst | «Referentielijst» | (UML) Datatype | Datatype |
| Referentie element | «Referentie element» | (UML) Property | Attribute |
| Codelist | «Codelist | (UML) Datatype | Datatype |

### 2.1.3 Overige

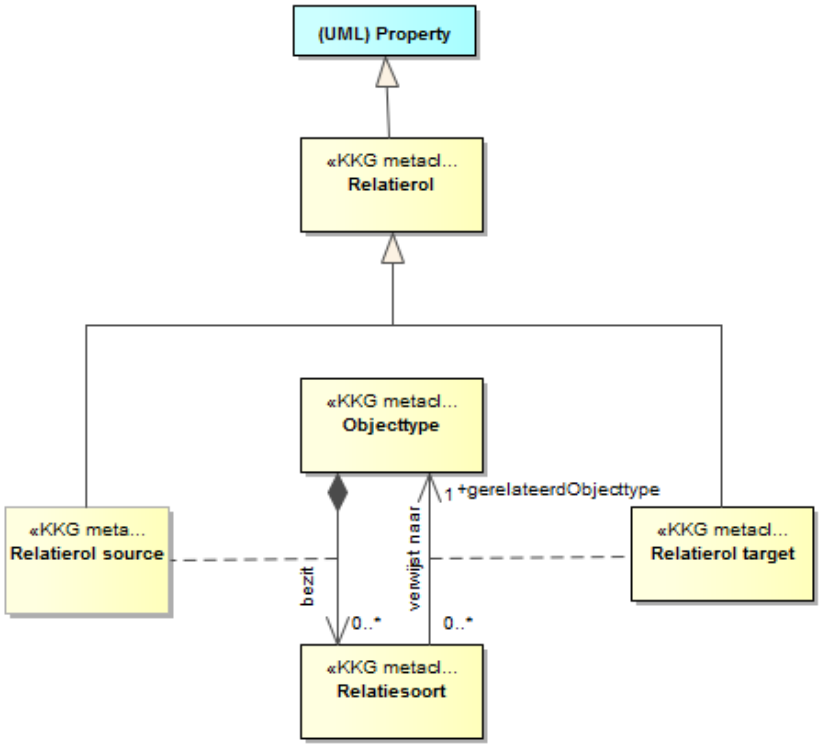
Constraint




**View 3: Constraint**

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Constraint | - | (UML) Constraint | Constraint |

Relatierol




**View 4: Relatiesoort en relatierol**

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Relatierol (abstract) | «Relatierol» | Property | Attribute |
| Relatierol source | «Relatierol» | Property | Attribute |
| Relatierol target | «Relatierol» | Property | Attribute |

Externe koppeling

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Externe koppeling | «Externe koppeling» | (UML) Association | Association |

Packages

| KKG metaclass | Stereotype | Metaclass UML 2.5 | In Sparx EA |
|---|---|---|---|
| Domein (het eigen IM) | - | (UML) Package | Package \| |
| Extern | «Extern» | (UML) Package | Package \| |
| View | «View» | (UML) Package | Package |

## 2.2 Betekenis modelelementen

In deze paragraaf staan alle modelelementen opgesomd, die gebruikt worden bij het maken van een informatiemodel. Bijna alle hebben een UML-metaclass als basis, deze is dan aangegeven. Dit is ook opgenomen in diagram vorm, in bijlage 3.

### 2.2.1 Objecten en attributen

1. **Objecttype** - *Stereotype «Objecttype»*: De UML-representatie van een objecttype, uitgedrukt in een stereotype van UML-Class (metaclass).

   Er zijn verschillende modelelementen die gebaseerd zijn op UML-Class, zoals aangegeven in §2.1. Wanneer een UML-Class in het informatiemodel gelezen moet worden als een objecttype, dan krijgt deze het stereotype «Objecttype».

   Een objecttype is een groep van gelijksoortige objecten. Om duidelijk te maken wat wordt bedoeld kijken we eerst naar het begrip 'object'.

   **Definitie Object**
   Een ding, een tastbaar iets, in de werkelijkheid, zoals daarnaar gekeken wordt vanuit een bepaald domein.

   *Toelichting*
   Het wordt veelal als niet politiek correct beschouwd mensen als objecten te zien. In dit kader, de informatievoorziening, beschouwen we evenwel natuurlijke en niet-natuurlijke personen wel als objecten. 'Tastbaar' moet hierbij ruim geïnterpreteerd worden. Het gaat niet alleen om fysiek herkenbare objecten zoals auto's, gebouwen en mensen, ook om zogenaamde virtuele objecten waarover binnen het domein door betrokkenen gecommuniceerd wordt zoals kadastrale percelen, (maatschappelijke) activiteiten en processen. Hoe een 'tastbaar iets' als een object beschouwd wordt, hangt af van het domein waarvoor dat 'tastbaar iets' relevant is. Zo wordt de gebouwde omgeving in het ene domein beschouwd als een verzameling gebouwen terwijl een ander domein daarin panden onderscheidt. Een object is voor een domein relevant als eigenschappen (kenmerken) daarvan van belang zijn voor het functioneren van dat domein.

   **Definitie Objecttype**
   De typering van een groep objecten (in de werkelijkheid) die binnen een domein relevant zijn en als gelijksoortig worden beschouwd.

   *Toelichting*
   Jan, Piet en Marie zijn mensen die vanuit het Burgerzaken-domein beschouwd worden als objecten van het type 'natuurlijk persoon'. In een ander domein, 'de volksmond', noemen we dit 'mens' wat ook een objecttype is. In weer een ander domein is Jan van het type 'vergunninghouder' en Piet en Marie niet, omdat aan hen (nog) nooit een vergunning verleend is. Objecttypen zijn een abstractie van de werkelijkheid oftewel we beogen hiermee de werkelijkheid zo getrouw mogelijk te beschrijven, binnen de context van het domein. Dit staat geheel los van het vastleggen van gegevens over objecten van een type in een registratie. Daartoe is veelal een interpretatie nodig (van die werkelijkheid cq. die objecttypen) naar eenheden die in een registratie vastgelegd kunnen worden (records, entiteiten e.d.) op basis van andere overwegingen.

2. **Attribuutsoort** – *Stereotype «Attribuutsoort»*: De UML-representatie van een attribuutsoort, uitgedrukt in een stereotype van UML-Property[3] (metaclass).

   Er zijn verschillende modelelementen die gebaseerd zijn op UML-property, zoals aangegeven in §2.1.2. Wanneer een UML-property in het informatiemodel de betekenis heeft van een attribuut van een objecttype, dan heeft deze het stereotype «Attribuutsoort».

   Een attribuutsoort is een type van gelijksoortige attributen of gegevens. Daartoe kijken we eerst naar het begrip 'gegeven'.

   **Definitie Gegeven**
   De betekenisvolle formulering van een waargenomen feit, waaraan een waarde kan worden toegekend.

   *Toelichting*
   Gegevens zijn de objectief waarneembare neerslag of registratie van feiten op een bepaald medium, zodanig dat deze gegevens uitgewisseld en voor langere tijd bewaard kunnen worden. Dat kan op papier, in digitale vorm, et cetera. Met deze gegevens wordt een model (een selectief deel dus) van de werkelijkheid vastgelegd in de tijd. Ofschoon de werkelijkheid nooit stilstaat, kan deze door het vastleggen van de gegevens toch worden bevroren.
   Voorbeelden van gegevens zijn de waardes 'Jan' en 'man' betreffende de naam en het geslacht van een object van het type Persoon. Merk op dat een gegeven zonder duidelijkheid over het type gegeven (naam, geslacht e.d.) geen informatie biedt. Een gegeven wordt ook wel attribuut genoemd.

   **Definitie Attribuutsoort**
   De typering van gelijksoortige gegevens die voor een objecttype van toepassing is.

   *Toelichting*
   Een gegeven met stereotype attribuutsoort is een kenmerk van een object.

   Attribuutsoorten worden ook wel kenmerken of eigenschappen genoemd. Aan elk objecttype worden nul, één of meer «Attribuutsoort»en toegekend. In een informatiemodel worden alleen voor het domein relevante attribuutsoorten opgenomen bij een objecttype.

   *Bijvoorbeeld: we kunnen definiëren dat 'persoonsnaam' en 'geslachtsaanduiding' attribuutsoorten zijn die van toepassing zijn voor het objecttype 'persoon'. Wanneer 'oogkleur' niet relevant is voor het domein, wordt deze niet gemodelleerd.*

3. **Gegevensgroep**– *Stereotype «Gegevensgroep»*: De UML-representatie van een gegevensgroep, uitgedrukt in een stereotype van UML-property (metaclass).

   **Definitie Gegevensgroep**
   Een typering van een groep van gelijksoortige gegevens die voor een objecttype van toepassing is.

   *Toelichting*
   Dit modelelement verzorgt de modelmatige aankoppeling van een gegevensgroeptype aan het objecttype waartoe een gegevensgroeptype onlosmakelijk behoort.

[3] In versies voor UML 2.5 heette deze nog UML-attribute.

De groep van gegevens is een kenmerk van een object. De gegevensgroep is een betekenisvol kenmerk van een objecttype. De gegevensgroep heeft altijd als type een gegevensgroeptype.

*Bijvoorbeeld: een persoon heeft ogen, dit is een onmiskenbaar kernmerk van een persoon. De gegevensgroep noemen we daarom 'ogen'. Elk oog heeft een kleur en een sterkte, waar we waardes van bij willen houden, zoals respectievelijk blauw en -2. Dit kan per oog verschillen, waardoor het nuttig kan zijn om deze kenmerken als attribuutsoorten onder te brengen in een gegevensgroeptype Oog. De gegevensgroep 'ogen' krijgt een definitie, en krijgt als type het Gegevensgroeptype Oog (merk op dat we het kenmerk Oog als modelelement kunnen definiëren, zonder dat er we er waardes van vastleggen. Alleen van kleur en sterkte leggen we waardes vast).*

4. **Gegevensgroeptype** – *Stereotype «Gegevensgroeptype»*: De UML-representatie van een gegevensgroeptype, uitgedrukt in een stereotype van UML-Class (metaclass).

**Definitie Gegevensgroeptype**
Een groep van met elkaar samenhangende attribuutsoorten. Een gegevensgroeptype is altijd een type van een gegevensgroep.

*Toelichting*
De attribuutsoorten van het gegevensgroeptype zijn semantisch gezien eigenschappen van het objecttype. Echter, vanwege samenhangend gedrag (ze horen semantisch bij elkaar, ze wijzigen bijvoorbeeld gelijktijdig e.d.) zijn deze ondergebracht in een apart modelelement. Het onderbrengen van attribuutsoorten in een groep c.q. in het modelelement gegevensgroeptype, is een keuze, het is niet perse noodzakelijk. Wanneer deze ondergebracht worden in een gegevensgroeptype dan zijn/blijven het afzonderlijke kenmerken van het object en dus attribuutsoorten van het objecttype, maar dan ondergebracht in een gegevensgroeptype. De gegevensgroep als geheel wordt daarom expliciet niet gezien als zijnde één attribuutsoort van een object.

*Bijvoorbeeld: Geboorte bij INGESCHREVEN NATUURLIJK PERSOON (met daarin onder andere de attribuutsoorten Geboortedatum en Geboortegemeente) en Motor (met daarin attribuutsoorten over de inhoud, vermogen, serienummer en het soort motor) bij een SCHIP.*

*Toelichting bij voorbeeld: in de BRK is een persoon eigenaar van een Schip, niet van een Motor. In de BRK kan het eigendom van een Motor niet worden overgedragen aan een ander persoon. In een ander informatiemodel, zoals van een motorfabriek, zou de Motor wel een objecttype kunnen zijn, omdat het daar wel hét onderwerp van gesprek is.*

Een gegevensgroeptype is meestal het type van slechts één gegevensgroep, omdat de semantiek meestal exclusief is voor één objecttype. Echter, hergebruik is mogelijk (als de semantiek niet exclusief is voor één objecttype). Voorwaarde voor hergebruik is dat de definitie (de definitie en toelichting, inclusief alle metadata aspecten) dan inderdaad gelijk zijn, voor alle objecttypes die hergebruik maken van het gegevensgroeptype.

Een gegevensgroeptype kan, naast attribuutsoorten, ook weer gegevensgroeptypen bevatten.

Een gegevensgroeptype is verbonden met een objecttype, via het modelelement Gegevensgroep.

### 2.2.2 Relaties

Verbanden met betekenis, die gelegd zijn tussen modelelementen van het type UML-Class.

5. **Generalisatie:** De UML-representatie van een specialisatie, uitgedrukt in een UML-generalization (metaclass).

   **Definitie Generalisatie**
   De typering van het hiërarchische verband tussen een meer generiek object van een objecttype en een meer specifiek object van een ander objecttype waarbij het laatstgenoemde object eigenschappen van het eerstgenoemde object overerft.

   *Toelichting*
   Een generalisatierelatie geeft aan dat bepaalde eigenschappen van een objecttype (vaak attribuutsoorten en/of relatiesoorten) ook gelden voor de gerelateerde objecttypen, én dat deze qua semantiek, structuur en syntax gelijk zijn. We spreken dan van een supertype met subtypen. De modelelementen die generiek gelden worden in een generiek objecttype, het supertype, gemodelleerd en deze worden overerft door elk subtype (minimaal twee) die de generalisatie relatie legt naar dit generieke objecttype.

   *Voorbeeld: PERCEEL is specialisatie van KADASTRAAL ONROERENDE ZAAK, APPARTEMENTSRECHT is specialisatie van KADASTRAAL ONROERENDE ZAAK. PERCEEL en APPARTEMENTSRECHT hebben beide 'Kadastrale aanduiding' en een 'relatie met ONROERENDE ZAAK FILIATIE'.*

6. **Relatiesoort** – *Stereotype «Relatiesoort»*: De UML-representatie van een relatiesoort, uitgedrukt in een stereotype van UML-association (metaclass).

   **Definitie Relatiesoort**
   De typering van het structurele verband tussen een object van een objecttype en een (ander) object van een ander (of hetzelfde) objecttype.

   *Toelichting*
   Objecten hebben eigenschappen die gemodelleerd kunnen worden met attribuutsoorten maar ook met relatiesoorten naar andere objecttypen. Als het voor het desbetreffende domein van belang is om die eigenschap te modelleren als onderdeel van een ander objecttype, dan maakt de relatiesoort die eigenschap beschikbaar voor het eerstgenoemde objecttype. Bijvoorbeeld, een attribuutsoort van het objecttype PERSOON zou kunnen zijn 'Naam geregistreerd partner' (naast de attribuutsoort 'Naam' van PERSOON). De naam van de geregistreerde partner komt evenwel ook beschikbaar met een relatiesoort van PERSOON naar PERSOON: "heeft geregistreerd partnerschap met". Zie ook het eerder genoemde voorbeeld van SCHIP en MOTOR.

   Voorbeeld: relatiesoorten "VERBLIJFSOBJECT is gelegen in een PAND" en "SUBJECT heeft als correspondentieadres WOONPLAATS", of korter, "gelegen in", "postadres".

   Wanneer een relatie (UML-assocation) gebruikt wordt om objecten aan elkaar te verbinden, zonder dat er eigenschappen over deze relatie worden vastgelegd, dan heeft deze het stereotype «Relatiesoort».

7. **Relatieklasse** - *Stereotype «Relatieklasse»*: De UML-representatie van een Relatieklasse, uitgedrukt in een stereotype van UML-associationClass (metaclass).

**Definitie Relatieklasse**
Een relatiesoort met eigenschappen.

*Toelichting*
De relatieklasse geeft aan dat er een relatie is tussen twee objecten, waarbij er gegevens over deze relatie vastgelegd moeten worden. De relatie wordt in dit geval behandeld als een object, met gegevens. De gegevens over de relatie bestaan alleen zolang de relatie tussen beide objecten bestaat en zolang elk van beide objecten zelf (nog) bestaan.

*Bijvoorbeeld: relatieklasse FUNCTIONARIS (een PERSOON is benoemd als de FUNCTIONARIS voor een AFDELING en heeft bijvoorbeeld een benoemingsdatum als attribuut).*

Opmerking: de gegevens van de relatie worden voor één relatie vastgelegd en niet voor meerdere relaties tegelijk. Als dit laatste het geval is, dan worden de gegevens vastgelegd in een «Objecttype». Een CONTRACT kan bijvoorbeeld ook een afspraak zijn tussen twee óf méér SUBJECTen, waarbij de gegevens van de relatie voor alle betrokken objecten hetzelfde zijn. CONTRACT wordt dan gemodelleerd als objecttype, waarbij beschreven wordt wat er moet gebeuren wanneer één van de SUBJECTen niet meer bestaat.

8. **Externe koppeling** – *Stereotype «Externe koppeling»***:** De UML-representatie van een externe koppeling, uitgedrukt in een stereotype van UML-association (metaclass). De source kant van het aggregatietype is 'composite' (de gesloten diamant staat aan de kant van het objecttype die de koppeling legt naar het externe objecttype).

   **Definitie Externe koppeling**
   Een associatie waarmee vanuit het perspectief van het eigen informatiemodel een objecttype uit het 'eigen' informatiemodel gekoppeld wordt aan een objecttype van een extern informatiemodel. De relatie zelf hoort bij het 'eigen' objecttype. Zie ook 3.14.

9. **Relatierol** – *Stereotype «Relatierol»*: De UML-representatie van een relatierol, uitgedrukt in een stereotype van UML-Property[4] (metaclass).

   **Definitie Relatierol**
   De benaming van de manier waarop een object deelneemt aan een relatie met een ander object.

   *Toelichting:*
   Met relatie wordt in deze elke relatie bedoeld een *«Relatiesoort»*, *«Relatieklasse»* of *«Externe koppeling»*. Voor «Generalisatie» speelt het niet. Een relatie heeft een source kant, die de eigenaar is van de relatie, en is gericht naar de target kant. De relatierol kan aan beide kanten een naam en een definitie krijgen.

   *Bijvoorbeeld: een kind-ouder relatie. Een PERSOON heeft in de rol 'ouder van' een relatie met PERSOON, 0, 1 of meerdere keren. Andersom is de rol 'kind van'. Een EIGENDOM kan overgedragen worden van de ene PERSOON naar de andere PERSOON. De relatie krijgt de naam 'overdracht' , met de source rol 'vervreemder' en de target rol 'verkrijger'.*

[4] In versies voor UML 2.5 werd de rol nog op UML-associationEnd gedefinieerd.

### 2.2.3. Waardenlijsten

Een datatype waarvan de mogelijke waarden zijn opgesomd in een lijst. De waarde van een attribuutsoort moet één van de waarden zijn uit de gespecificeerde waardenlijst.

10. **Referentielijst** – *Stereotype «Referentielijst»***:** De UML-representatie van een referentielijst, uitgedrukt in een stereotype van UML-Datatype (metaclass).

    **Definitie Referentielijst**
    Een lijst met een opsomming van de mogelijke domeinwaarden van een attribuutsoort, die buiten het model in een externe waardenlijst worden beheerd. De domeinwaarden in de lijst kunnen in de loop van de tijd aangepast, uitgebreid, of verwijderd worden, zonder dat het informatiemodel aangepast wordt (in tegenstelling tot bij een enumeratie).

    *Bijvoorbeeld: referentielijst LAND, referentielijst NATIONALITEIT. Een NATUURLIJK PERSOON heeft een attribuut geboorteland, van het type LAND.*

    De referentielijst bevat representaties van objecten, die in het informatiemodel niet als een objecttype onderwerp van gesprek zijn. De referentielijst wordt gebruikt als type van een attribuut van een object.

    Het objecttype LAND uit het voorbeeld is opgenomen in een referentielijst en niet als objecttype. Maar we willen wel de structuur en betekenis van LAND vastleggen, zodat we er naar kunnen refereren. Een object dat is opgenomen in een referentielijst heeft daarom veelal meerdere attributen, zoals de naam, de ontstaansdatum, een omschrijving en de ISO code, die zijn opgenomen in de referentie lijst.

    Alle attributen van gerefereerde objecten uit de referentielijst gelden in de context van het informatiemodel, mits opgenomen in de *«Referentielijst»*. In de registratie wordt vaak alleen de referentie ernaartoe opgenomen, omdat het niet de bedoeling is om alle gegevens over te nemen. De gegevens staan immers al in de referentielijst en er is bewust gekozen om een referentielijst te modelleren. Het attribuut van een objecttype dat als type een referentielijst heeft bevat in de registratie daarom (vaak) alleen een referentie naar een object uit de lijst.

11. **Referentie element** – *Stereotype «Referentie element»*: De UML-representatie van een referentie-element uitgedrukt in een stereotype van UML-Property(metaclass).

    **Definitie**
    Een eigenschap van een object in een referentielijst in de vorm van een gegeven.

    *Bijvoorbeeld: referentielijst LAND kent de referentie-elementen Landcode ISO en Landnaam, referentielijst NATIONALITEIT kent referentie-element Nationaliteitcode.*

    Een referentie element kan uniek zijn, zoals een code, en is dan op zichzelf geschikt om gebruikt te worden als referentie (zoals bedoeld in de definitie van Referentielijst).

12. **Enumeratie**
    Voor enumeraties is geen stereotype gespecificeerd. In het metamodel maken we gebruik van de bestaande UML-enumeration (metaclass) voor de specificaties van een enumeratie.

    **Definitie Enumeratie**
    Een datatype waarvan de mogelijke waarden limitatief zijn opgesomd in een statische lijst.

    *Toelichting*
    In de registratie krijgt een attribuut één van deze waarden. De lijst is een statische lijst met constanten (meerdere attributen, zoals bij een referentielijst, zijn nooit aan de orde).

    *Bijvoorbeeld: geslacht (man, vrouw, overig), de typering van een openbare ruimte (spoorbaan, plein, straat).*

13. **Enumeratiewaarde**
    Voor enumeratiewaarde is geen stereotype gespecificeerd. In het metamodel maken we gebruik van de bestaande UML-enumerationLiteral (metaclass) voor de specificaties van een enumeratiewaarde.

    **Definitie Enumeratiewaarde**
    Een gedefinieerde waarde, in de vorm van een eenmalig vastgesteld constant gegeven.

    *Bijvoorbeeld: geslacht: man, vrouw, overig, type openbare ruimte: spoorbaan, plein, straat.*

14. **Stereotype** «**Codelist**»: De UML-representatie van een codelist, uitgedrukt in een stereotype van UML-datatype (metaclass).

    **Definitie Codelist**
    De definitie van een codelist is gelijk aan de definitie van een referentielijst.

    Er is wel een verschil in modellering; zie hiervoor de toelichting.

    *Bijvoorbeeld: codelist LAND met daarin (alleen) het attribuut 'naam' (de externe gepubliceerde waardenlijst bevat naast de naam ook de ISO code en de ontstaansdatum).*

    *Toelichting*
    Zowel referentielijsten als codelists zijn in feite waardenlijsten. In tegenstelling echter tot de referentielijst wordt een codelist *niet* in het informatiemodel beschreven, omdat de definitie en semantiek geheel in de externe waardenlijst staat en niet (nader) geduid hoeft te worden in het informatiemodel zelf. Een codelist heeft in het informatiemodel daarom geen attributen (en zou voor de definitie alleen hoeven te refereren naar de definitie bij de extern gepubliceerde waardenlijst, maar voor het gemak is de definitie wel opgenomen als metagegeven in dit metamodel). De extern gepubliceerde waardenlijst bevat, naast gewone attributen, ook altijd één specifiek attribuut, met daarin de domeinwaarden die gebruikt mogen/moeten worden in de registratie. In het gebruik is een Codelist daarom analoog aan een Enumeratie. Welk specifiek attribuut dit is en wat de betekenis daarvan is staat in de codelist zelf gedefinieerd.

### 2.2.4 Datatypen

Een datatype die de structuur beschrijft waaraan een waarde (zie 2.2.1 Objecten en attributen) moet voldoen.

Bij elke «Attribuutsoort» wordt gespecificeerd aan welk datatype de data c.q. de waarde die hiervoor vastgelegd wordt moet voldoen. Het datatype wordt gebruikt als type van een attribuutsoort.

Anders gezegd, Datatypes zijn veelal herbruikbaar en kunnen gespecificeerd worden bij diverse «Attribuutsoort»-en.

15. **Primitief datatype -** «Primitief datatype»: in het metamodel maken we gebruik van de bestaande UML-PrimitiveType (metaclass) voor de specificaties van een primitief datatype.

    **UML-PrimitiveType**
    Een standaard datatype, zoals bekend in vele specificatietalen, dat de structuur van een gegeven beschrijft. Het metamodel volgt waar mogelijk de definities zoals beschreven in ISO standaarden (zie §3.1). Deze datatypes hebben altijd al een naam en definitie gekregen vanuit deze standaarden en deze worden gebruikt. Deze worden niet door de modelleur gecreëerd en hebben daarom geen KKG metaclass.

    *Voorbeeld: CharacterString, Integer, DateTime.*

    **Definitie Primitief datatype**
    Een in het eigen model gedefinieerd primitieve datatype. Deze worden wel door de modelleur gecreëerd, met een eigen naam en een eigen definitie (en eigen metagegevens).

    *Voorbeeld: Documentnummer, Postcode. In het geval van Postcode is de landelijke definitie in tekst vastgelegd buiten het informatiemodel zelf, waarbij in het eigen model een modelelement is gemaakt in de vorm van het datatype Postcode[5].*

    *Toelichting*
    Een primitief datatype is een datatype zonder verdere specificatie over de structuur. Dit datatype kent geen UML-property en dus ook geen elementen met stereotype «Data element». Dit datatype is enkelvoudig, oftewel niet samengesteld, en wordt ook wel simpel datatype genoemd.

    Wanneer een Primitief datatype wordt gespecificeerd, dan heeft deze standaard als primitive datatype een CharacterString.

    Een informatiemodel definieert datatypes als er behoefte is aan een datatype dat eenmalig gedefinieerd wordt en op meerdere plekken in het model gebruikt moet kunnen worden met altijd exact dezelfde structuur en waardenbereik (zie ook ‘patroon’ in 3.5). Dit datatype, met een eigen naam, wordt vervolgens gebruikt als type van een attribuutsoort.

[5] Opmerking: *w*anneer het datatype Postcode landelijk zodanig beschikbaar is gemaakt zodat hier gebruik van gemaakt kan worden in het model, dan hoeft Postcode niet meer in het eigen model opgenomen te worden.

16. **Gestructureerd datatype** – *stereotype «Gestructureerd datatype»***:** De UML-representatie van een gestructureerd datatype uitgedrukt in UML-datatype (metaclass) met ten minste twee keer een UML-Property.

    **Definitie Gestructureerd datatype**
    Specifiek benoemd gestructureerd datatype dat de structuur van een gegeven beschrijft, samengesteld uit minimaal twee elementen.

    *Toelichting*
    In UML wordt een Gestructureerd datatype een structured Datatype genoemd.

    De waarde van het attribuutsoort verkoopprijs met datatype bedrag is uitgedrukt in een combinatie van een som en valuta zoals 35 euro. De introductie van één datatype Bedrag, uitgedrukt in som en valuta, legt dus vast dat som en valuta onlosmakelijk met elkaar zijn verbonden.

    De eigenschappen in het Gestructureerd datatype tezamen zijn identificerend (een Gestructureerd datatype "identificeert zichzelf", zoals er maar per definitie één "1 liter" bestaat, één 35 euro en één datum 6 april 2017, met per definitie altijd dezelfde betekenis:
    - Een blik olie heeft een inhoud van **7 liter**, kost **35 euro**, en is verkocht op **6 april 2017**.
    - Piet heeft **1 liter** bloed gedoneerd, daarvoor **35 euro** vergoeding gekregen, op **6 april 2017**.

    Het identificerend zijn geldt bijvoorbeeld niet voor Jan Jansen. Er zijn meerdere personen met deze naam en dat zijn verschillende personen (Jan Jansen is dan ook een gegevensgroeptype Naam met voornaam Jan en achternaam Jansen en geen Gestructureerd datatype).

    *Voorbeeld: Gestructureerd datatype Bedrag bestaat uit de data-elementen som en valuta.*

17. **Data element** - *Stereotype «Data element»***:** De UML-representatie van een data element uitgedrukt in UML-property (metaclass).

    **Definitie Data element**
    Een onderdeel/element van een Gestructureerd datatype die als type een datatype heeft.

    *Toelichting*
    Het data element is een eigenschap van een Gestructureerd datatype en beschrijft de structuur van een gegeven. Het is niet een eigenschap van een object en niet hetzelfde als een attribuutsoort.

    Het data element beschrijft in combinatie met andere data-elementen de structuur van een gegeven en heeft zelf een datatype. Dit datatype is meestal een primitief datatype.

18. **Union** – *Stereotype «Union»***:** De UML-representatie van een union uitgedrukt in UML-datatype (metaclass).

    **Definitie Union**
    Gestructureerd datatype, waarmee wordt aangegeven dat er meer dan één mogelijkheid is voor het datatype van een attribuut. Het attribuut zelf krijgt als datatype de union. De union biedt een keuze uit verschillende datatypes, elk afzonderlijk beschreven in een union element.

    *Voorbeeld: Union LineOrPolygon. Deze biedt een keuze uit Union element Line of Union element Polygon.*

19. **Union element** - *Stereotype «Union element»***:** De UML-representatie van een union element uitgedrukt in UML-property (metaclass), dat zelf een type heeft dat uitgedrukt is in een UML-datatype (metaclass).

    **Definitie Union element**
    Een type dat gebruikt kan worden voor het attribuut zoals beschreven in de definitie van Union. Het union element is een onderdeel van een Union, uitgedrukt in een eigenschap (attribute) van een union, die als keuze binnen de Union is gerepresenteerd..

    *Voorbeeld: union element Line en union element Polygon zijn beiden onderdeel van Union LineOrPolygon*

### 2.2.5 Packages

Een package is een benoemde en begrensde verzameling/groepering van modelelementen.

20. **Extern** - Stereotype «Extern»**:** De UML-representatie van een extern package uitgedrukt in UML-package (metaclass)

    **Definitie Extern**
    Een groepering van constructies die een externe instantie beheert en beschikbaar stelt aan een informatiemodel en die in het informatiemodel ongewijzigd gebruikt worden.

    *Voorbeeld: het Externe package NEN3610 met datatype NEN3610ID. Het datatype van attribuutsoort Identificatie wegdeel in RSGB verwijst naar het datatype NEN3610ID zoals opgenomen in het Externe package.*

21. **View** - *Stereotype «View»***:** De UML-representatie van een view package uitgedrukt in UML-package (metaclass)

    **Definitie View**
    Een groepering van objecttypen die gespecificeerd zijn in een extern informatiemodel en vanuit het perspectief van het eigen informatiemodel inzicht geeft welke gegevens van deze objecttypen relevant zijn binnen het eigen informatiemodel.

    *Bijvoorbeeld: IMKAD-BRP. Een aantal van de gegevens uit BAG objecten uit de basisregistratie BAG zijn relevant voor de basisregistratie Kadaster. De definities van de BAG worden gevolgd. Vanuit modelleringsperspectief wordt dit gezien als een view.*

### 2.2.6 Overig

22. **Id** - *Stereotype «id»*[6] *bij target role van de «relatiesoort»*: De UML-representatie van een relatie die een identificerende rol speelt, uitgedrukt bij een UML-property (metaclass).

    **Definitie**
    Aanduiding dat de relatiesoort waarop de *«id»* is gedefinieerd een onderdeel is van de unieke aanduiding van een objecttype.

    Toelichting:
    Dit wordt alleen gedaan voor objecttypes die zelf geen unieke aanduiding hebben en daarom deze moeten samenstellen met de unieke aanduiding van het gerelateerde objecttype.

    *Voorbeeld: BUURT heeft zelf geen unieke identificatie. Een BUURT ligt in een WIJK en binnen die WIJK is de BUURT wel uniek. De WIJK zelf heeft een unieke identificatie. De unieke identificatie van BUURT is daarom samengesteld uit het attribuut Buurtcode van BUURT en de verwijzing naar de WIJK (de identificatie van WIJK).*

23. **Constraint** - Voor Constraint is geen stereotype gespecificeerd. In het metamodel maken we gebruik van de bestaande UML-Constraint (metaclass).

    **Definitie**
    Een constraint is een conditie of een beperking, die over een of meerdere modelelementen uit het informatiemodel geldt.

    *Toelichting*
    Een constraint kan vastgelegd worden bij alle modelelementen die als basis een UML-metaclass hebben waarvan UML aangeeft dat hier een UML-constraint op gedefinieerd mag worden. Aanbeveling is om dit waar mogelijk op een *«Objecttype»* te doen of eventueel (indien van toepassing) op een *«Gegevensgroeptype»* of *«Relatieklasse».*
    Een constraint wordt altijd in gewone tekst omschreven en optioneel als formele specificatie in de Object Constraint Language (OCL). Dit is verder uitgewerkt in Hoofdstuk 3.

    *Bijvoorbeeld: een conditionele afhankelijkheid 'als (optioneel) attribuut 1 leeg is, dan is (optioneel) attribuut 2 verplicht', of een bijzondere regel, zoals '11-proef is van toepassing op dit attribuut'.*

[6] In UML 2.5 heeft een target role een UML-Property waarop **isID** kan worden gespecificeerd. In Enterprise Architect nog niet. Daarom wordt er tijdelijk gewerkt met dit stereotype. Op termijn komt deze te vervallen.

## 2.3 Specificatie metagegevens

Elk modelelement kent een aantal metagegevens, die bepaalde aspecten van het modelelement specificeren. Een aantal daarvan worden gemodelleerd in UML. Deze zijn herkenbaar aan de rode tekst (hiervan worden geen waardes vastgelegd). Een aantal worden als waarde vastgelegd, in tagged value vastgelegd. Deze zijn herkenbaar aan *Tagged value.* Deze tagged values zijn specifiek voor elk modelelement (als er in H2.2 sprake is van een generalisatie, dan worden deze tagged values niet overerft. De KKG metaclass Union erft geen metagegevens, zoals patroon, van KKG metaclass Datatype).

### 2.3.1 Specificatie metagegevens voor objecten en attributen

**Specificatie voor «Objecttype»[7]**

De objecttypen worden naar de volgende aspecten gespecificeerd:

| Aspect[8] | Kardinaliteit | Toelichting | In UML 2.5[7] | In EA[9] |
|---|---|---|---|---|
| **Naam√*** | 1 | De naam van het objecttype[10]. | *name van de metaclass Named element* | *Name* |
| **Herkomst*** | 1 | De registratie in wiens catalogus het objecttype is gespecificeerd (oftewel de registratie waar het objecttype deel van uitmaakt). Deze specificatie is toegevoegd omdat het wel duidelijk moet zijn in welke (basis)registratie of informatiemodel het objecttype voorkomt (indien van toepassing). | | *Tagged value* |
| **Definitie√*** | 1 | De beschrijving van de betekenis van het objecttype zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie of informatiemodel. | *Body van de metaclass Comment* | *Notes* |
| **Herkomst definitie√** | 1 | Voor objecttypen die deel uitmaken van een registratie is de definitie hieruit overgenomen. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop het objecttype is opgenomen in het informatiemodel. | | *Tagged value* |

[7] In deze kolom is opgenomen hoe het element in UML2.5 is benoemd. Het betreft veelal overerving van een gegeven van een UML metaclass die niet in dit document is benoemd.

[8] *Aspect met aanduiding √ is conform stelselafspraken voor basisregistraties. Een * is conform de stelselcatalogus. Die ook de paragraaf in H3 hierover.*

[9] *Rode tekst in de kolom 'In EA' betreft een standaard element binnen Sparx EA. Zwarte tekst in de kolom 'in EA' betreft uitbreiding op UML Metamodel, via tagged values of aanvullende stereotypes.*

[10] *In (basis) registraties is dit meestal gespecificeerd in een catalogus van objecten en begrippen. Deze opmerking geldt voor elk metadata aspect naam van de andere modelelementen. Indien het modelelement niet voorkomt in een dergelijke catalogus is dan kiest u uiteraard een eigen naam.*

| Aspect[8] | Kardinaliteit | Toelichting | In UML 2.5[7] | In EA[9] |
|---|---|---|---|---|
| **Unieke aanduiding**√ | 1 | Voor objecttypen die deel uitmaken van een (basis)registratie of informatiemodel betreft dit de wijze waarop daarin voorkomende objecten (van dit type) uniek in de registratie worden aangeduid. | - | *isId bij attribuutsoort, --- of --- stereotype «isId» bij target role relatiesoort --- of --- een combinatie van deze twee, elk hiervan meer keren toepasbaar* |
| **Populatie**√* | 0..1 | Voor objecttypen die deel uitmaken van een (basis)registratie betreft dit de beschrijving van de exemplaren van het gedefinieerde objecttype die in de desbetreffende (basis)-registratie voorhanden zijn. | | *Tagged value* |
| **Kwaliteit**√* | 0..1 | Voor objecttypen die deel uitmaken van een registratie betreft dit de waarborgen voor de juistheid van de in de registratie opgenomen objecten van het desbetreffende type. | | *Tagged value* |
| **Toelichting**√* | 0..1 | Voor objecttypen die deel uitmaken van een (basis)registratie of informatiemodel betreft dit de daarin opgenomen toelichting. | | *Tagged value* |
| **Indicatie abstract object** | 1 | Conceptueel model: indicatie dat het objecttype een generalisatie is, waarvan een object als specialisatie altijd voorkomt in de hoedanigheid van een (en slechts één) van de specialisaties van het betreffende objecttype.<br><br>Logisch model: Indicatie dat er geen instanties (objecten) voor het betreffende objecttype mogen voorkomen. | *isAbstract bij de metaclass Classifier* | *Abstract* |

**Specificatie voor «Attribuutsoort»**

De attribuutsoorten worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5[7] | In EA |
|---|---|---|---|---|
| **Naam**√* | 1 | De naam van de attribuutsoort. | *name van de metaclass Named element* | *Name* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan de attribuutsoort ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Definitie**√* | 1 | De beschrijving van de betekenis van de attribuutsoort. | *Body van de metaclass Comment* | *Notes* |
| **Herkomst definitie**√ | 1 | De registratie of het informatiemodel waaruit de definitie is overgenomen dan wel een aanduiding die aangeeft uit welke bronnen de definitie is samengesteld. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop de attribuutsoort is opgenomen in het informatiemodel. | | *Tagged value* |
| **Domein** *(aspecten van een waarde/data)* | | *Domein is zelf geen metadata aspect. Onder het kopje 'domein' vallen een aantal metadata aspecten die gelden voor een waarde, oftewel de eisen waaraan een waarde van een attribuutsoort moet voldoen.* | | |
| - **Type**√ | 1 | Het type waarmee waarden van deze attribuutsoort worden vastgelegd. Dit is altijd conform een datatype uit dit metamodel (of een extensie ervan).<br><br>Betreft het een waarde uit een dynamische waardentabel, dan wordt de naam van de desbetreffende referentielijst of codelist als type vermeld.<br>Indien het een waarde uit een statische opsomming van waarden betreft, dan wordt de naam van de desbetreffende enumeratie als type vermeld. | | *Type* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5[7] | In EA |
|---|---|---|---|---|
| - **Lengte** | 0..1 | De aanduiding van de lengte van een gegeven. Getallen kunnen altijd positief of negatief zijn.<br><br>*Bijvoorbeeld:*<br>*'1' als de lengte exact 1 is;*<br>*'1..2' als de lengte 1 tot en met 2 lang kan zijn;*<br>*''1,2' voor Decimale getallen met 1 cijfer voor de komma en 2 erna.* Dit is van -9,99 tot +9,99;<br><br>Dit is verder toegelicht in hoofdstuk 3. | | *Tagged value* |
| - **Patroon** | 0..1 | Alleen van toepassing wanneer het type van het attribuutsoort een primitief datatype is.<br><br>De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben, dat wil zeggen het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| - **Formeel patroon** | 0..1 | Alleen van toepassing wanneer het type van het attribuutsoort een primitief datatype is.<br><br>Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Indicatie materiële historie**√ | 1 | Indicatie of de materiële historie van de attribuutsoort te bevragen is. Materiële historie geeft aan wanneer een verandering is opgetreden in de werkelijkheid die heeft geleid tot verandering van de attribuutwaarde. | | *Tagged value* |
| **Indicatie formele historie**√ | 1 | Indicatie of de formele historie van de attribuutsoort te bevragen is. Formele historie geeft aan wanneer in de administratie een verandering is verwerkt van de attribuutwaarde (wanneer was de verandering bekend en is deze verwerkt). | | *Tagged value* |
| **Locatie** | 0..1 | Als het type van het attribuutsoort een waardenlijst is, dan wordt hier de locatie waar deze te vinden is opgegeven. Dit is in principe een URI (als er geen URI is, dan kan dit een URL zijn, waar de waardenlijst op basis van de naam van de waardenlijst te vinden is). | | *taggged value* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5[7] | In EA |
|---|---|---|---|---|
| **Kardinaliteit**√ | 1 | Deze indicatie geeft aan hoeveel keer waarden van deze attribuutsoort kunnen voorkomen bij een object van het betreffende objecttype, of bij het betreffende gegevensgroeptype:<br>0..1: is soms niet beschikbaar<br>1 : is altijd beschikbaar<br>0..*: is niet altijd beschikbaar, kan meerdere malen voorkomen<br>1..*: is altijd beschikbaar, kan meerdere malen voorkomen<br><br>Indien een attribuutsoort deel uit maakt van een gegevensgroeptype, dan wordt de kardinaliteit vermeld van het attribuutsoort binnen het gegevensgroeptype. Voor de uiteindelijke kardinaliteit van hoe vaak een gegeven voorkomt bij het object moet rekening gehouden worden met de kardinaliteit van de gegevensgroep en met de kardinaliteit van de attribuutsoort. | *lowerValue en upperValue van de metaclass Multiplicity Element* | *Multiplicity* |
| **Authentiek**√ * | 1 | Aanduiding of het een authentiek gegeven (attribuutsoort) betreft. | | *Tagged value* |
| **Toelichting**√ • | 0..1 | Een inhoudelijke toelichting op de attribuutsoort. | | *Tagged value* |
| **Indicatie afleidbaar** | 1 | Aanduiding dat gegeven afleidbaar is uit andere attribuut- en/of relatiesoorten. | *isDerived bij metaclass Property* | *isDerived* |
| **Mogelijk geen waarde** | 1 | Aanduiding dat attribuutsoort geen waarde kan bevatten (de waarde zou er kunnen zijn, maar kan ook onbekend zijn, of bewust weggelaten). | | *Tagged value* |
| **Identificerend** | 0..1 | Aanduiding dat attribuutsoort onderdeel uitmaakt van de unieke aanduiding van een object | *isID bij de metaclass Property* | *isID* |

**Specificatie voor «Gegevensgroep»**

De gegevensgroepen worden naar de volgende aspecten gespecificeerd:

| | | | | |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het gegevensgroep. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 1 | De beschrijving van de betekenis van de gegevensgroep. | *Body van de metaclass Comment* | *Notes* |
| **Toelichting** | 0..1 | Een inhoudelijke toelichting op de gegevensgroep. | | *Tagged value* |
| **Gegevensgroeptype** | 1 | De verwijzing naar het bijbehorende gegevensgroeptype. | | *Type* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan het gegevensgroep ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Herkomst definitie** | 1 | De registratie of het informatiemodel waaruit de definitie is overgenomen dan wel een aanduiding die aangeeft uit welke bronnen de definitie is samengesteld. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop het gegevensgroep is opgenomen in het informatiemodel. | | *Tagged value* |
| **Indicatie materiële historie** | 1 | Indicatie of de materiële historie van het gegevensgroep te bevragen is. Materiële historie geeft aan wanneer een verandering is opgetreden in de werkelijkheid die heeft geleid tot verandering van de attribuutwaarde. | | *Tagged value* |
| **Indicatie formele historie** | 1 | Indicatie of de formele historie van het gegevensgroep te bevragen is. Formele historie geeft aan wanneer in de administratie een verandering een van de attribuutwaarden van de attribuutsoorten in de groep is verwerkt (wanneer was de verandering bekend en is deze verwerkt). | | *Tagged value* |

| | | | | |
|---|---|---|---|---|
| **Kardinaliteit** | 1 | Deze indicatie geeft aan hoeveel keer de gegevensgroep kan voorkomen bij een object van het betreffende objecttype:.<br>0..1: is soms niet beschikbaar<br>1 : is altijd beschikbaar<br>0..*: is niet altijd beschikbaar, kan een opsomming zijn<br>1..*: is altijd beschikbaar, kan een opsomming zijn.<br><br>Indien een attribuutsoort deel uit maakt van een gegevensgroeptype, dan wordt de kardinaliteit vermeld van het attribuutsoort binnen het gegevensgroeptype. Voor de uiteindelijke kardinaliteit van hoe vaak een gegeven voorkomt bij het object moet rekening gehouden worden met de kardinaliteit van de gegevensgroep en met de kardinaliteit van de attribuutsoort. | *lowerValue en upperValue van de metaclass Multiplicity Element* | *Multiplicity van de source role van de bijbehoren de composite relatie* |
| **Authentiek** | 1 | Aanduiding of het een authentiek gegeven betreft. | | *Tagged value* |

**Specificatie voor «Gegevensgroeptype»**
De gegevensgroeptypen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het gegevensgroeptype. | *name van de metaclass Named element* | *Name* |
| **Datum opname** | 1 | De datum waarop het gegevensgroeptype is opgenomen in het informatiemodel. | | *Tagged value* |

### 2.3.2 Specificatie metagegevens voor relaties

**Relatiesoort en relatierol**

**Het metamodel heeft twee manieren om een relatie tussen twee objecttypen te beschrijven. Deze keuze wordt aangegeven in de eigen extensie, zoals beschreven in paragraaf 1.8. Alleen het gekozen alternatief is relevant voor de modellering in uw informatiemodel.**

- **Alternatief 1: Verplichte benoeming van de naam van de relatie met de bijbehorende metagegevens**
- **Alternatief 2: Verplichte benoeming van de rol van de target in een relatie met de bijbehorende metagegevens en optioneel de benoeming van de naam van de relatie.**

Beide alternatieven gebruiken relatiesoort en relatierol, maar met andere regels voor gebruik.

#### 2.3.2.1 Relatiesoort leidend (alternatief 1)

Relatiesoort is verplicht, met een naam en met een definitie en deze is leidend. Metadata aspecten worden hierbij altijd vastgelegd. Het gebruik van relatierol is optioneel (zowel bij source en target). Áls er een relatierol target wordt vastgelegd, dan is de metadata hierbij wel verplicht.

**Specificatie voor «Relatiesoort»**

De relatiesoorten worden naar de volgende aspecten gespecificeerd.

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam**√ | 1 | De naam van de relatiesoort. | *name van metaclass Named element* | *Name* |
| **Uni-directioneel** | 1 | Het gerelateerde objecttype (de target) waarvan het objecttype, die de eigenaar is van deze relatie (de source), kennis heeft.<br><br>Alle relaties zijn altijd gericht van het objecttype (source) naar het gerelateerde objecttype (target). | | *Direction van de betreffende assiciation (van source naar target)* |
| **Objecttype** | 1 | Het objecttype waarvan de relatie een eigenschap is. | */source: related Element bij Relationship →Element* | *Source* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Gerelateerd objecttype** | 1 | Het objecttype waarmee een objecttype een logisch verband heeft | */target: relatedElement bij Relationship →Element* | *Target* |
| **Type aggregatie** | 1 | Standaard betreft het geen aggregatie (None).<br><br>Het type aggregatie mag 'composite' zijn. Dit wordt gedaan als er een afhankelijkheid is in die zin dat de target niet kan bestaan zonder de source: de target vervalt als de source vervalt. | *isComposite bij metaclass Property* | *Aggregation van de source role met waarde composite* |
| **Kardinaliteit**√ | 1 | Deze indicatie geeft aan hoeveel keer waarden van deze relatiesoort (i.c. relaties) kunnen voorkomen bij een object van het betreffende objecttype:<br>0..1: is soms niet beschikbaar<br>1 : is altijd beschikbaar<br>0..*: is niet altijd beschikbaar, kunnen meerdere relaties zijn<br>1..*: is altijd beschikbaar, kunnen meerdere relaties zijn<br>*..*: is niet altijd beschikbaar, kunnen meerdere relaties zijn tussen objecten van hetzelfde objecttype.<br>Indien een relatiesoort deel uit maakt van een gegevensgroeptype, dan wordt de kardinaliteit vermeld van de relatiesoort binnen het gegevensgroeptype. Voor de uiteindelijke kardinaliteit van de relatiesoort moet ook rekening gehouden worden met de kardinaliteit van het gegevensgroeptype. | *lowerValue en upperValue van de metaclass MultiplicityElement* | *Multiplicity van de target role* |
| **Herkomst** * | 1 | De registratie of het informatiemodel waaraan de relatiesoort ontleend is, dan wel de eigen organisatie. Indien zelf toegevoegd, dan is de herkomst de eigen organisatie. | | *Tagged value* |
| **Definitie**√ * | 1 | De beschrijving van de betekenis van de relatiesoort. Deze is verplicht als er geen source role respectievelijk target role is gespecificeerd. | *Body van de metaclass Comment* | *Notes* |
| **Herkomst definitie**√ | 1 | De registratie of het informatiemodel waaruit de definitie is overgenomen dan wel een aanduiding die aangeeft uit welke bronnen de definitie is samengesteld. | | *Tagged value* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Datum opname** | 1 | De datum waarop de relatiesoort is opgenomen in het informatiemodel. | | *Tagged value* |
| **Indicatie materiële historie**√ | 1 | Indicatie of de materiële historie van de relatiesoort te bevragen is. Materiële historie geeft aan wanneer een verandering is opgetreden in de werkelijkheid die heeft geleid tot verandering van de relatie. | | *Tagged value* |
| **Indicatie formele historie**√ | 1 | Indicatie of de formele historie van de relatiesoort te bevragen is. Formele historie geeft aan wanneer in de administratie een verandering is verwerkt van de relatie (wanneer was de verandering bekend en is deze verwerkt). | | *Tagged value* |
| **Authentiek**√* | 1 | Aanduiding of de attribuutsoort waarvan de relatiesoort is afgeleid, een authentiek gegeven (attribuutsoort) betreft. | | *Tagged value* |
| **Indicatie afleidbaar** | 1 | Aanduiding dat gegeven afleidbaar is uit andere attribuut- en/of relatiesoorten. | *isDerived bij UML metaclass Assocation* | *isDerived* |
| **Toelichting**√* | 0..1 | Een inhoudelijke toelichting op de relatiesoort. | | *Tagged value* |
| **Mogelijk geen waarde** | 1 | Aanduiding dat relatiesoort geen waarde met betekenis kan bevatten. | | *Tagged value* |

### Specificatie voor «Relatierol»

Voor relatierollen worden naar de volgende aspecten gespecificeerd.

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 0..1 | De naam van de relatiesoort. | *name van de metaclass Namedelement* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van dit type | *Body van de metaclass Comment* | *Notes* |

### 2.3.2.2 Relatierol is leidend (alternatief 2)

Verplichte benoeming van de rol van de target in een relatie met de bijbehoren de metagegevens en optioneel de benoeming van de naam van de relatie.

**Specificatie voor «Relatiesoort»**

De relatiesoorten worden naar de volgende aspecten gespecificeerd.

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 0..1 | De naam van de relatiesoort. | *name van de metaclass Namedelement* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van de relatiesoort. | *Body van de metaclass Comment* | *Notes* |

**Specificatie voor «Relatierol»**
Voor relatierol worden bij de target rol van een relatiesoort de volgende aspecten gespecificeerd.

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de relatierol<br><br>Bijvoorbeeld: eigenaar bij een eigendomsrelatie. | *name van de metaclass Named element* | *Name* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan de relatierol ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Definitie√*** | 1 | De beschrijving van de betekenis van de relatierol. | *Body van de metaclass Comment* | *Notes* |
| **Herkomst definitie√** | 1 | De registratie of het informatiemodel waaruit de definitie is overgenomen dan wel een aanduiding die aangeeft uit welke bronnen de definitie is samengesteld. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop de relatierol is opgenomen in het informatiemodel. | | *Tagged value* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Kardinaliteit**√ | 1 | Deze indicatie geeft aan hoeveel keer waarden van deze relatierol kunnen voorkomen. | *lowerValue en upperValue van de metaclass Multiplicity Element* | *Multiplicity* |
| **Indicatie materiële historie**√ | 1 | Indicatie of de materiële historie van de relatierol te bevragen is. Materiële historie geeft aan wanneer een verandering is opgetreden in de werkelijkheid die heeft geleid tot verandering van de waarde van de relatierol. | | *Tagged value* |
| **Indicatie formele historie**√ | 1 | Indicatie of de formele historie van de relatierolte bevragen is. Formele historie geeft aan wanneer in de administratie een verandering is verwerkt van de waarde van de relatierol (wanneer was de verandering bekend en is deze verwerkt). | | *Tagged value* |
| **Authentiek**√* | 1 | Aanduiding of het een authentiek gegeven (relatierol) betreft. | | *Tagged value* |
| **Mogelijk geen waarde** | 1 | Aanduiding dat relatierol geen waarde met betekenis kan bevatten. | | *Tagged value* |
| **Toelichting**√* | 0..1 | Een inhoudelijke toelichting op de relatierol. | | *Tagged value* |

### Specificatie voor «Generalisatie»

De generalisaties worden naar het volgende aspect gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 0..1 | De naam van de generalisatie. Standaard 'is specialisatie van'. | *name van de metaclass Named element* | *Name* |
| **Objecttype** | 1 | Het objecttype dat een specialisatie is van een (ander) objecttype. | */source: related Element bij Relationship →Element* | *Source* |
| **Gerelateerd objecttype** | 1 | Het objecttype dat de generalisatie is van een (ander) objecttype. | */target: related Element bij Relationship →Element* | *Target* |

**Specificatie voor «Relatieklasse»**
De relatieklassen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de relatieklasse. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 1 | De beschrijving van de betekenis van de relatieklasse | *Body van de metaclass Comment* | *Notes* |

**Specificatie voor «Externe koppeling»**
Externe koppelingen worden naar de volgende aspecten gespecificeerd.

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 0..1 | De naam van de externe koppeling. Standaard 'betreft'. | *name van de metaclass Named element* | *Name* |
| **Datum opname** | 1 | De datum waarop de externe koppeling is opgenomen in het informatiemodel. | | *Tagged value* |
| **Objecttype** | 1 | Het objecttype waarvan de relatie een eigenschap is. | */source: related Element bij Relationship →Element* | *Source* |
| **Type aggregatie** | 1 | Aanduiding dat het een compositie relatie is. Waarde is altijd Composite. | *isComposite van Property* | *Aggregation in de Source role* |
| **Gerelateerd objecttype** | 1 | Het objecttype uit een extern informatiemodel waarmee een objecttype een logische verbinding heeft. | */target: related Element bij Relationship →Element* | *Target* |
| **Uni-directioneel** | 1 | Het gerelateerde objecttpe uit een extern informatiemodel (de target) waarvan het objecttype die de eigenaar van deze relatie is (de source) kennis heeft. Het aggregation type van de source is altijd 'composition'.<br><br>Alle relaties zijn altijd gericht van het objecttype (source) naar het gerelateerde objecttype (target). | | *Direction (van source naar target)* |

### 2.3.3. Specificatie metagegevens voor waardenlijsten

**Specificatie voor «Referentielijst»**

Voor referentielijsten worden de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de lijst. | *name van de metaclass Namedelement* | *Name* |
| **Herkomst** | 1 | De registratie in wiens catalogus de lijst is gespecificeerd (oftewel de registratie waar de referentielijst deel van uitmaakt).[11] | | *Tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van de lijst zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *Body van de metaclass Comment* | *Notes* |
| **Datum opname** | 1 | De datum waarop de lijst is opgenomen in het informatiemodel. | | *Tagged value* |
| **Toelichting** | 0..1 | Voor lijsten die deel uitmaken van een registratie betreft dit de daarin opgenomen toelichting. | | *Tagged value* |

**Specificatie voor «Referentie element»**

De referentie-elementen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het referentie-element. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 1 | De beschrijving van de betekenis van het referentie-element. | *Body van de metaclass Comment* | *Notes* |
| **Datum opname** | 1 | De datum waarop het referentie-element is opgenomen in het informatiemodel. | | *Tagged value* |

[11] Deze specificatie is toegevoegd t.o.v. de registratiecatalogus aangezien het hier niet om een registratie gaat maar wel duidelijk moet zijn in welke registratie de (verwijzing naar de) lijst voorkomt (indien van toepassing).

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Domein** *(aspecten van een waarde/data)* | | | | |
| - **Type** | 1 | Het type waarmee gegevens/waarden van deze attribuutsoort worden vastgelegd. Dit is altijd conform een datatype uit dit metamodel (of een extensie ervan).<br><br>Betreft het een waarde uit een dynamische waardentabel, dan wordt de naam van de desbetreffende referentielijst of codelist als type vermeld. Indien het een waarde uit een statische opsomming van waarden betreft, dan wordt de naam van de desbetreffende enumeratie als type vermeld. | | *Type* |
| - **Lengte** | 0..1 | De aanduiding van de lengte van een gegeven. Getallen kunnen altijd positief of negatief zijn.<br><br>*Bijvoorbeeld:*<br>*'1' als de lengte exact 1 is;*<br>*'1..2' als de lengte 1 tot en met 2 lang kan zijn;*<br>*"1,2' voor getallen met 1 cijfer voor de komma en 2 erna.* Dit is van -9,99 tot +9,99;<br>Dit is verder toegelicht in hoofdstuk 3. | | *Tagged value* |
| - **Patroon** | 0..1 | De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben oftewel het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| - **Formeel patroon** | 0..1 | Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Kardinaliteit** | 1 | Deze indicatie geeft aan hoeveel keer waarden van dit referentie-element kunnen voorkomen bij een refentielijst van het betreffende type:<br>0..1: is soms niet beschikbaar<br>1 : is altijd beschikbaar<br>0..* is niet altijd beschikbaar, kan een opsomming zijn<br>1..*: is altijd beschikbaar, kan een opsomming zijn. | *lowerValue en upperValue van de metaclass Multiplicity Element* | *Multiplicity van de de target role* |
| **Identificatie** | 0..1 | Aanduiding dat referentie-element onderdeel uitmaakt van de unieke aanduiding van een referentielijst. | *isID van de metaclass Property* | *isID bij de betreffende class* |

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Toelichting** | 0..1 | Een inhoudelijke toelichting op het referentie-element. | | *Tagged value* |

**Specificatie voor «codeList»**

Voor codelist worden de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de lijst zoals gespecificeerd in de catalogus van de desbetreffende registratie dan wel, indien het een door de eigen organisatie toegevoegde lijst betreft, de door de eigen organisatie vastgestelde naam. | *name van de metaclass Named element* | *Name* |
| **Herkomst** | 1 | De registratie in wiens catalogus de lijst is gespecificeerd (oftewel de registratie waar de lijst deel van uitmaakt). [11] | | *tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van de lijst zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *Body van de metaclass Comment* | *Notes* |
| **Datum opname** | 1 | De datum waarop de lijst is opgenomen in het informatiemodel. | | *tagged value* |
| **Toelichting** | 0..1 | Voor lijsten die deel uitmaken van een registratie betreft dit de daarin opgenomen toelichting. | | *tagged value* |
| **Locatie** | 1..1 | Als het type van het attribuutsoort een waardenlijst is, dan wordt hier de locatie waar deze te vinden is opgegeven. Indien mogelijk is de verwijzing een URI of een URL (als er geen URI is, dan kan dit een URL zijn, waar de waardenlijst op basis van de naam van de waardenlijst te vinden is). | | *tagged value* |

### 2.3.4. Specificatie metagegevens voor datatypen

**Specificatie voor «Datatype»**
De datatypen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het data-element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van het data-element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *Body van de metaclass Comment* | *Notes* |
| **Domein** *(aspecten van een waarde/data)* | | | | |
| **- Lengte** | 0..1 | De aanduiding van de lengte van een gegeven. Getallen kunnen altijd positief of negatief zijn.<br><br>*Bijvoorbeeld:*<br>*'1' als de lengte exact 1 is;*<br>*'1..2' als de lengte 1 tot en met 2 lang kan zijn;*<br>*''1,2' voor Decimale getallen met 1 cijfer voor de komma en 2 erna.* Dit is van -9,99 tot +9,99;<br><br>Dit is verder toegelicht in hoofdstuk 3. | | *Tagged value* |
| - **Patroon** | 0..1 | De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben, oftewel het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| - **Formeel patroon** | 0..1 | Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan het Gestructureerd datatype ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop het Gestructureerd datatype is opgenomen in het informatiemodel. | | *Tagged value* |

**Specificatie voor «Gestructureerd datatype»**

Voor Gestructureerde datatypen worden de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het Gestructureerd datatype zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *name van de metaclass Named element* | *Name* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan het Gestructureerd datatype ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van het Gestructureerd datatype zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *Body van de metaclass Comment* | *Notes* |
| **Patroon** | 0..1 | De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben, oftewel het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| **Formeel patroon** | 0..1 | Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Datum opname** | 1 | De datum waarop het Gestructureerd datatype is opgenomen in het informatiemodel. | | *Tagged value* |

**Specificatie voor «Data element»**

De data-elementen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het data-element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van het data-element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *Body van de metaclass Comment* | *Notes* |

| | | | | |
|---|---|---|---|---|
| **Domein**<br>*(aspecten van een waarde/data)* | | | | |
| - **Type** | 1 | Het type waarmee waarden van deze attribuutsoort worden vastgelegd. Dit is altijd conform een datatype uit dit metamodel (of een extensie ervan).<br><br>Betreft het een waarde uit een dynamische waardentabel, dan wordt de naam van de desbetreffende referentielijst of codelist als type vermeld.<br>Indien het een waarde uit een statische opsomming van waarden betreft, dan wordt de naam van de desbetreffende enumeratie als type vermeld. | | *Type* |
| **- Lengte** | 0..1 | De aanduiding van de lengte van een gegeven. Getallen kunnen altijd positief of negatief zijn.<br><br>*Bijvoorbeeld:*<br>*'1' als de lengte exact 1 is;*<br>*'1..2' als de lengte 1 tot en met 2 lang kan zijn;*<br>*''1,2' voor Decimale getallen met 1 cijfer voor de komma en 2 erna.* Dit is van -9,99 tot +9,99;<br><br>Dit is verder toegelicht in hoofdstuk 3. | | *Tagged value* |
| - **Patroon** | 0..1 | De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben, oftewel het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| - **Formeel patroon** | 0..1 | Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Kardinaliteit** | 1 | Deze indicatie geeft aan hoeveel keer waarden van dit data-element kunnen voorkomen bij een refentielijst van het betreffende type:<br>0..1: is soms niet beschikbaar<br>1 : is altijd beschikbaar<br>0..* is niet altijd beschikbaar, kan een opsomming zijn<br>1..*: is altijd beschikbaar, kan een opsomming zijn. | *lowerValue en upperValue van de metaclass Multiplicity Element* | *Multiplicity* |

**Specificatie voor «Union»**

De unions worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de union zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *name van de metaclass Namedelement* | *Name* |
| **Herkomst** | 1 | De registratie of het informatiemodel waaraan de union ontleend is dan wel de eigen organisatie indien het door de eigen organisatie toegevoegd is. | | *Tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van de union zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *Body van de metaclass Comment* | *Notes* |
| **Datum opname** | 1 | De datum waarop de union is opgenomen in het informatiemodel. | | *Tagged value* |

### 2.3.15 Specificatie voor «Union element»

De unionelementen worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het union element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van het union element zoals gespecificeerd in de catalogus van de desbetreffende (basis)registratie. | *Body van de metaclass Comment* | *Notes* |

| | | | | | |
|---|---|---|---|---|---|
| **Domein**<br>*(aspecten van een waarde/data)* | | | | | |
| | | | | | *Type* |
| - | **Type** | 1 | Het type waarmee gegevens van deze attribuutsoort worden vastgelegd. Dit is altijd conform een datatype uit dit metamodel (of een extensie ervan).<br><br>Betreft het een waarde uit een dynamische waardentabel, dan wordt de naam van de desbetreffende referentielijst of codelist als type vermeld.<br>Indien het een waarde uit een statische opsomming van waarden betreft, dan wordt de naam van de desbetreffende enumeratie als type vermeld. | | |
| - | **Lengte** | 0..1 | De aanduiding van de lengte van een gegeven. Getallen kunnen altijd positief of negatief zijn.<br><br>*Bijvoorbeeld:*<br>*'1' als de lengte exact 1 is;*<br>*'1..2' als de lengte 1 tot en met 2 lang kan zijn;*<br>*"1,2' voor Decimale getallen met 1 cijfer voor de komma en 2 erna.* Dit is van -9,99 tot +9,99;<br><br>Dit is verder toegelicht in hoofdstuk 3. | | *Tagged value* |
| - | **Patroon** | 0..1 | De verzameling van waarden die gegevens van deze attribuutsoort kunnen hebben, oftewel het waardenbereik, uitgedrukt in een specifieke structuur. | | *Tagged value* |
| - | **Formeel patroon** | 0..1 | Zoals patroon, formeel vastgelegd (met een reguliere expressie), uitgedrukt in een formele taal die door de computer wordt herkend. | | *Tagged value* |
| **Kardinaliteit** | | 1 | Deze indicatie geeft aan hoeveel keer waarden van dit unionelement kunnen voorkomen bij een referentielijst van het betreffende type.<br>De kardinaliteit van een unionelement is altijd 1. | *lowerValue en upperValue van de metaclass MultiplicityElement* | *iplicity* |

### 2.3.5 Specificatie metagegevens voor packages

**Specificatie voor «Extern»**

Externe packages worden naar de volgende aspecten gespecificeerd:

| Aspect | | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het externe package zoals gespecificeerd door de externe instantie. | *name van de metaclass Namedelement* | *Name* |
| **Locatie** | 1 | De verwijzing naar de locatie van het bijbehorende package (dit kan een geheel model zijn in één package). Indien mogelijk is de verwijzing een URI of een URL. | | *Tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van het package, gezien vanuit het eigen informatiemodel.<br><br>*Bijvoorbeeld: bron van definities.* | *Body van de metaclass Comment* | *Notes* |
| **Herkomst** * | 1 | De registratie of het informatiemodel waaraan het package ontleend is. Bij een view is de herkomst nooit de eigen organsiatie. | | *Tagged value* |

**Specificatie voor «View»**

View packages worden naar de volgende aspecten gespecificeerd, analoog aan «Extern»:

| Aspect | | Toelichting | In UML 2.5 | In EA? |
|---|---|---|---|---|
| **Naam** | 1 | De naam van het view package.<br><br>Deze is, indien mogelijk, analoog aan de naamgeving in het externe schema waar de view over gaat, eventueel met een prefix. | *name van de metaclass Named element* | *Name* |
| **Locatie** | 1 | De verwijzing naar de locatie van het bijbehorende informatiemodel waar de view over gaat. Indien mogelijk is de verwijzing een URI of een URL. | | *Tagged value* |
| **Definitie** | 1 | De beschrijving van de betekenis van het package. | *Body van de metaclass Comment* | *Notes* |
| **Herkomst** * | 1 | De registratie of het informatiemodel waaraan het package ontleend is. Bij een view is de herkomst nooit de eigen organsiatie. | | *Tagged value* |

### 2.3.6 Specificatie metagegevens - overig

**Specificatie voor Enumeratie(waarden)**

Enumeraties betreffen de metaclass Enumeration en worden naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de enumeratie zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 1 | De beschrijving van de betekenis van de enumeratie zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *Body van de metaclass Comment* | *Notes* |

**Specificatie voor Enumeratie(waarden)**

De enumeratiewaarde zelf betreft de metaclass UML-EnumerationLiteral en kent volgende aspecten:

| Aspect | | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam** | 1 | De naam van de enumeratiewaarde zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *name van de metaclass Named element* | *Name* |
| **Definitie** | 0..1 | De beschrijving van de betekenis van de enumeratiewaarde zoals gespecificeerd in de catalogus van de desbetreffende registratie. | *Body van de metaclass Comment* | *Notes* |
| **Code** | 0..1 | De in een registratie of informatiemodel aan de enumeratiewaarde toegekend unieke code (niet te verwarren met alias, zoals bedoeld in 2.6.1). | *Alias van de metaclass Element Import* | *Alias* |

**Specificatie voor een Constraint**

Constraint betreft de metaclass UML Constraint en wordt naar de volgende aspecten gespecificeerd:

| Aspect | Kardinaliteit | Toelichting | In UML 2.5 | In EA |
|---|---|---|---|---|
| **Naam**√ | 1 | De naam van de constraint. | *name van de metaclass Named element* | *Name* |
| **Specificatie tekst** | 0..1 | De specificatie van de constraint in normale tekst. | | *Notes (type = invariant)* |
| **Specificatie formeel** | 0..1 | De beschrijving van de constraint in een formele specificatietaal, in OCL | | *Notes (type = OCL)* |

#### 2.3.6.1 Alias

Alle metamodel elementen hebben een naam (metadata aspect) en hiervoor is het mogelijk om naast de naam ook een alternatieve weergave van de naam op te nemen. Deze wordt gemodelleerd met een alias. De alias is optioneel (zie ook 3.16).

De specificatie van de alias is overgenomen uit UML. De alias is te specificeren voor NamedElements (zoals UML-Class, UML-association, UML-UML-Datatype en UML-property). De alias is in UML gespecificeerd bij de metaclass Element Import[12]. In Enterprise Architect is de alternatieve weergave aan te zetten in de properties van een Diagram, via: use alias if available.

Een uitzondering is gemaakt voor UML-EnumerationLiteral. De 'naam' betreft hier een daadwerkelijk waarde, waarin de naam gelijk staat aan de waarde. Het is daarom expliciet ongewenst om hiervoor een alternatieve naamgeving te gebruiken. De alias wordt hier, mede daarom, gebruikt voor (alleen) de modellering van het metadata aspect Code, welke aanvullend is op naam (niet een alternatief van naam).

#### 2.3.6.2 Tagged values en waardenbereik tagged values

Tagged values, zoals genoemd in de UML-extensie kolom zijn altijd van het datatype CharacterString. Aanvullend geldt:

- Voor lengtes geldt dat er alleen getallen in mogen (van het datatype Integer).
- Voor datums geldt dat deze het volgende patroon volgen: jjjjmmdd

| **Tagged value** | **Waardenbereik** |
|---|---|
| Indicatie materiële historie | Ja, Nee, zie Groep |
| Indicatie formele historie | Ja, Nee, zie Groep |
| Mogelijk geen waarde | Ja, Nee |
| Authentiek[13] | Authentiek, Basisgegeven, Landelijk kerngegeven, Gemeentelijk kerngegeven, Overig[14] |

## 2.4 Metamodel Tooling

Er is een metamodel *profiel* gemaakt in Sparx Enterprise Architect, dat gebruikt kan worden bij het modelleren van een informatiemodel. Dit profiel kan je inladen en daarna kan je kiezen uit de metamodel elementen. Het profiel is faciliterend en zorgt dat (de meeste) modelelementen van het informatiemodel automatisch voldoen aan dit metamodel. Het is niet vereist om dit profiel te gebruiken. Het is niet toegestaan om het profiel te wijzigen. Het is wel toegestaan om het profiel uit te breiden, naar de behoefte van de eigen organisatie.

Er is een tool *Imvertor*, die kan controleren of een informatiemodel voldoet aan dit metamodel en zo niet, wat de reden daarvan is. Deze tool is open source en is te vinden op www.imvertor.org.

[12] Element import wordt in UML ingezet voor het importeren van een NamedElement uit een ander package. In dit metamodel wordt de alias (nog) niet zo gebruikt.
[13] Voor toelichting, zie paragraaf 3.11
[14] Geef bij overig in uw eigen informatiemodel aan wat u er onder verstaat.

# 3. Afspraken & Regels

In deze paragraaf gaan we in op een aantal aspecten van het zojuist beschreven metamodel en afspraken en regels die van toepassing zijn voor een informatiemodel.

## 3.1 Datatype(n)

Een datatype is een typering van een eigenschap. Datatypen in een informatiemodel beschrijven de structuur waaraan de data van objecten moet voldoen. De essentie van datatypen is dat ze gaan over de waarden die attribuutsoorten kunnen hebben. Het specificeert de waarden die de attribuutsoort kan aannemen en de vorm waarin deze beschikbaar zijn.

We onderscheiden de volgende soorten gedefinieerde categorieën voor datatypen:

1. Datatypen, primitief: data zoals "Amersfoort" of "10" worden vastgelegd als CharacterString en Integer. Dit volgt de internationale standaarden voor datatypen.
2. Datatypen (met een naam), landelijk volgens het GAB: datatypen zoals Postcode "1234AB".
3. Gestructureerde datatypen: een combinatie van data, zoals een bedrag "5 euro", of een GM_Surface. Deze volgen ook internationale of nationale standaarden.

De primitieve datatypen uit onderstaande lijst moeten gebruikt worden. De landelijke en de Gestructureerde datatypen hoeven niet per sé gebruikt te worden, maar het gebruik hiervan wordt wel aanbevolen. Dit metamodel voorziet er vooral in dat dit soort datatypen gedefinieerd kunnen worden, conform de modellering zoals het metamodel aangeeft, maar dus zonder de specifieke datatypen voor te schrijven.

Datatypen die niet in deze paragraaf (sub paragrafen) voor-gedefinieerd zijn worden in het eigen informatiemodel toegevoegd. Dit kan een organisatie zelf doen, door deze expliciet te beschrijven in de eigen extensie en daarna te gebruiken in het eigen informatiemodel. Deze gelden dan alleen voor het eigen informatiemodel.

Het is mogelijk om specifieke(re) restricties, zoals de lengte, toe te voegen. Dit wordt gedaan in een metagegeven lengte. De data van het attribuut moet dan voldoen aan het datatype én aan het metagegeven lengte. De lengte wordt dus niet in het datatype zelf vastgelegd.

### 3.1.1 Primitive datatypes

Dit metamodel onderkend (momenteel) de volgende extern gedefinieerde primitive datatypes. Deze zijn allemaal gebaseerd op [GAB]:

| **Primitive type** | **Betekenis** |
|---|---|
| CharacterString | Zie ISO 19103. Vrij vertaald: alle alfanumerieke tekens en speciale tekens die horen bij de gekozen characterset (standaard UTF-8), dus met diakrieten, white spaces, \-teken en newlines of HTML opmaak e.d. Mag starten met spatie. De maximale lengte is onbepaald.<br><br>Opmerking: getallen (ISO Numbers) met voorloopnullen worden opgenomen als CharacterString, met een patroon of formeel patroon. Bij het metagegeven Waardenverzameling attribuutsoort wordt dit dan (ook) gespecificeerd. |
| Integer | Zie ISO11404 (subtype van ISO Number). |

| Primitive type | Betekenis |
|---|---|
| | Vrij vertaald: geheel getal, lengte is minimaal 1 en maximale lengte is onbepaald, zonder voorloopnullen.<br>Opmerking: t.a.v. positieve en negatieve getalen en + en – tekens: bijvoorbeeld -2,0<br>Het (formeel) patroon geeft aan of een + en/of - teken gebruikt mag worden in het gegeven. |
| Real | Zie ISO11404 (subtype van ISO Number).<br>Vrij vertaald: een reëel getal, oftewel een rationeel getal zoals een Integer of een Decimal, of niet rationeel getal, zoals pi of de wortel van 2. Deze bestaat uit een (oneindig) aantal getallen, al dan niet achter de komma (floating point).<br>Opmerking: t.a.v. positieve en negatieve getalen en + en – tekens: zie Integer. |
| Boolean | Indicatie met mogelijke waarden True, false, 1 of 0. True en 1 hebben een identieke betekenis: Ja. False en 0 hebben een identieke betekenis: Nee.<br>Opmerking: t.a.v. Ja of Nee. Wanneer u de Ja of Nee wilt gebruiken, gebruik dan bv. een Enumeratie genaamd Indicatie, of gebruik AN met een lengte en een (formeel) patroon. |
| | |
| Date | 4-cijferig jaar, 2-cijferig maand, 2-cijferig dag uitgedrukt in yyyy-mm-dd conform https://en.wikipedia.org/wiki/ISO_8601 |
| DateTime | yyyy-mm-ddThh:mm:ss conform https://en.wikipedia.org/wiki/ISO_8601 |
| Year | 4-cijferig jaar uitgedrukt in yyyy conform https://en.wikipedia.org/wiki/ISO_8601 |
| Day | 2-cijferige dag uitgedrukt in dd conform https://en.wikipedia.org/wiki/ISO_8601 |
| Month | 2-cijferige maand uitgedrukt in mm conform https://en.wikipedia.org/wiki/ISO_8601 |
| | |
| URI | Unieke identificatie op internet conform RFC3986 en de URI-strategie Linked Open Data. Gestandaardiseerde manier om op het internet dingen (pagina's met informatie, objecten, datasets) uniek te identificeren. |

Het is mogelijk om in de eigen extensie extra primitive datatypes op te nemen, zodat deze ok beschikbaar komen voor het informatiemodel.

**Getallen en negatieve getallen**
Met een getal kan worden gerekend. Bijvoorbeeld: saldo, hoeveelheid, aantal, grootte.

Een getal kan negatief zijn. Dit is inherent aan een getal. Het - of + teken heeft geen invloed op de lengte van het getal. Als er dus een lengte wordt gespecificeerd, tel het - of + teken dan niet mee. Als een getal niet negatief mag zijn, geef dit dan aan in een patroon (zie NonNegativeInteger).

Aanbeveling: als er niet mee gerekend kan worden, zoals de bankrekening zelf (waar een saldo op staat, maar die bedoelen we hier expliciet niet), gebruik dan een CharacterString met een patroon.

### 3.1.2. Primitief datatype zelf definiëren

Het is ook mogelijk om in het eigen informatiemodel een eigen primitive datatype te definiëren in de vorm van een «Primitief datatype», met als UML metaclass de UML-Primitive datatype.

***Voorbeelden*** hiervan, die niet tot KKG behoren, maar ter illustratie zijn opgenomen, zijn:

- NietNegatieveInteger: een Integer die alleen de waarde 0 of groter mag hebben. Laat de naam van het primitieve type dan wel terugkomen in de naam (dus niet NietNegatiefGetal).
- Een beperking op een Real te specificeren door Decimal op te nemen (een gebroken getal, met (één of meer) cijfers voor de komma en cijfers achter de komma, conform ISO11404).
- AN. Deze is gebruikelijk bij een aantal basisregistraties. Datatype met een eigen naam, analoog aan CharacterString, maar met alleen 'normale' tekens. Dit zijn alle alfanumerieke tekens (dus inclusief diakrieten), de koppeltekens – en _ en spaties. De minimale lengte is tenminste 1, de maximale lengte is onbepaald. De 1e positie mag géén spatie bevatten.

### 3.1.3. Datatypen landelijk

Wanneer op landelijk niveau afspraken zijn gemaakt (bijvoorbeeld in GAB), voor algemene datatypen, die niet primitief zijn, zoals Postcode, dan worden deze niet in dit KKG metamodel opgenomen. Wel geeft KKG aan hoe deze in informatiemodellen gedefinieerd en gemodelleerd worden.

***Voorbeelden*** hiervan van landelijke datatypen, die niet tot KKG behoren, maar ter illustratie zijn opgenomen:

| | |
|---|---|
| Postcode | De in Nederland gangbare postcode voor een Nederlands postadres, bestaande uit een numeriek deel en een alfabetisch deel. Het numerieke deel van de postcode bestaat uit vier cijfers, het alfabetische deel van de postcode bestaat uit twee hoofdletters. Conform [GAB Postcodes]. |
| DMO | Datum mogelijk onvolledig. De keuze (<<union>>) van een periode in de Gregoriaanse kalender, al naar gelang de beschikbare datumelementen, uit de onderliggende subformaten alleen Year, Year en Month of Year, Month en Day. Dit is (nog steeds) overeenkomstig met https://en.wikipedia.org/wiki/ISO_8601 en [GAB DatumMogelijkOnvolledig]. |
| DTMO | Een volledige datum waarbij (alleen) de tijd mogelijk ontbreekt. De tijd wordt, zover bekend, ingevuld. Dus alleen de uren als de minuten onbekend zijn.<br>- DateTime, als de tijd wel volledig bekend is<br>- Date, als alleen de Date bekend is |

## 3.2 Gestructureerd datatype

Een «Gestructureerd datatype» is veelal specifiek binnen een informatiemodel. Indien mogelijk wordt zoveel mogelijk hergebruik gemaakt van elders gedefinieerde «Gestructureerd datatype»n, denk bijvoorbeeld aan de Gestructureerd datatypen: NEN3610 identificatie (NEN3610), Kadastrale aanduiding (BRK), Objectnummering (BAG) of Labelpositie.

Gewone datatypen staan op zichzelf en worden niet beschreven in termen van een ander datatype. Bij een «Gestructureerd datatype» is dit wel het geval. Het is een gestructureerd datatype dat is samengesteld uit

meerdere eigenschappen. Hiermee kunnen meerdere data-elementen, die onlosmakelijk bij elkaar horen, ook bij elkaar gedefinieerd worden.

Bijvoorbeeld een Bedrag, dat bestaat uit een hoeveelheid en een muntsoort. Het aantal zelf is nietszeggend, tenzij ook aangegeven wordt welke muntsoort het betreft.

Elk data-element in een Gestructureerd datatype heeft zelf ook weer een datatype (in zeer bijzondere gevallen kan een data-element zelf ook weer een Gestructureerd datatype zijn).

*Gestructureerd datatype representeren als één gegevenselement*
Soms is er de behoefte om een combinatie van gegevens samengesteld te representeren, in één gegevenselement. Dit speelt specifiek bij gestructureerde datatypes, omdat de data-elementen hiervan uniek identificerend zijn voor een object. De samengestelde representatie verandert niets aan de semantische definitie. Om een uniforme samenstelling te waarborgen, wordt er bij het gestructureerde datatype een patroon of een formeel patroon gedefinieerd (dat consistent is met de definities van de data-elementen uit het Gestructureerd datatype). Als een patroon of formeel patroon gedefinieerd is op het gestructureerde datatype (als geheel), dan worden de data-elementen van het gestructureerde datatype altijd als één gegevenselement uitgewisseld. Als dit patroon niet gedefinieerd is, dan worden de data-elementen als losse gegevenselementen uitgewisseld (de standaardwijze voor een Gestructureerd datatype).

Een uitgewerkt voorbeeld van een Gestructureerd datatype met patroon is Objectnummering. Dit Gestructureerd datatype bestaat uit de data elementen:

- Gemeentecode (AN, lengte 4)
- Objecttypecode (AN, lengte 2)
- Nummer (AN, lengte 10)

met daarbij een formeel patroon: [0-9]{4}\.[0-9]{2}\.[ 0-9]{10} of een (tekst) patroon Gemeentecode.Objecttypecode.Nummer

## 3.3 Gegevensgroeptype

Bij het modelleren van een objecttype worden attribuutsoorten toegekend aan een objecttype. Wanneer er geconstateerd wordt dat een aantal attribuutsoorten logisch gezien bij elkaar horen, dan kan er gekozen worden om deze onder te brengen in een gegevensgroeptype. Het blijven dan attributen van het objecttype. De definitie van elk afzonderlijk attribuutsoort verandert niet door de groepering.

Richtlijn: een gegevensgroeptype bevat alleen «Attribuutsoort»en en geen «Relatiesoort»en. Als er wel een relatie aan de orde is, dan wordt de gegevensgroep gezien als een apart te beheren object. Er wordt dan een apart «Objecttype» gemaakt. Het is wel mogelijk, hoewel uitzonderlijk, om binnen een gegevensgroeptype nog een ander gegevensgroeptype te modelleren.

### 3.3.1. Hergebruik

Het kan voorkomen dat meerdere objecttypes gebruik maken van dezelfde gegevensgroeptype, omdat de definitie voor alle objecttypes gelijk is of moet zijn. Het is dan modelmatig mogelijk om zo'n gegevensgroeptype te hergebruiken  Op conceptueel niveau is dit ongebruikelijk, omdat  kenmerken / eigenschappen  exclusief bij een object horen, maar het is wel toegestaan. Het is hierbij belangrijk om goed te kijken of er echt sprake is van een gegevensgroeptype, of dat er (toch) sprake is van een objecttype. Bijvoorbeeld van een objecttype die voor zijn bestaan afhankelijk is van een ander objecttype (en daarom via een «Relatiesoort» met aggregatietype 'composite' (het gesloten wiebertje) gekoppeld moet worden). Het is aan de modelleur om deze beoordeling te maken.

*Voorbeeld: de specificatie voor het Brondocument in de basisregistratie BAG is qua betekenis en structuur voor alle objecttypes gelijk. Het speelt een belangrijke rol in het totstandkomingsproces van objecten en behoort daarom in het conceptuele model. Het wordt ingezet als een audittrail, zodat het duidelijk op basis van welk brondocument een wijziging is doorgevoerd op een object.  Het brondocument onderscheidt twee relevante attribuutsoorten, maar het brondocument wordt binnen de BAG niet gezien als een gespreksonderwerp waarover men gegevens wilt communiceren. Daarom is er gekozen voor een gegevensgroeptype. Deze wordt hergebruikt voor alle objecttypes.*

Metamodel: het gegevensgroeptype kan dus het type zijn van meerdere gegevensgroepen. Vanwege dit hergebruik is daarom de kardinaliteit van de relatie van gegevensgroep naar gegevensgroeptype aan de source kant 1..*. Zie 2.1.1.

### 3.3.2 Gegevensgroep versus Gestructureerd datatype

Een gegevensgroep is niet hetzelfde als een Gestructureerd datatype.

- Een datatype beschrijft de structuur van data, een gegevensgroep beschrijft de semantiek van een kenmerk van een object.
- Als één kenmerk van een object uit verschillende stukjes data bestaat, dan wordt een Gestructureerd datatype gebruikt. Dit is bijvoorbeeld het geval bij het gestructureerde datatype Bedrag. Deze bestaat uit een 'hoeveelheid' en 'muntsoort.
- Als een object meerdere kenmerken heeft, gemodelleerd als afzonderlijke attribuutsoorten, dan heeft elk kenmerk op zichzelf betekenis. Als één kenmerk wordt weggelaten, of niet bekend of ingewonnen is, dan verandert er niets aan de betekenis van de andere attribuutsoorten.
- Een ander goed criterium is: als het datatype wordt weggelaten uit het informatiemodel, dan verliest het informatiemodel geen semantiek. Alleen de structuur van een gegeven is dan niet meer bekend.

Regel: het is niet de bedoeling dat eenzelfde kenmerk van een object in het ene model als een gegevensgroep gemodelleerd wordt en in het andere model als een Gestructureerd datatype. Er raakt dan semantiek verloren of er ontbreekt dan semantiek.

- Een Gestructureerd datatype in een conceptueel informatiemodel is en blijft dus altijd ook een Gestructureerd datatype in een bijbehorende logisch informatiemodel. Een gegevensgroep in een conceptueel model is en blijft dus altijd ook een gegevensgroep in een logisch informatiemodel.

## 3.4 Keuze tussen datatypes (Union)

Wanneer het datatype van een attribuutsoort een keuze uit twee of meer datatypen is, dan wordt dit gemodelleerd met het datatype Union. Elk union element van de union heeft dan één datatype, de waarde van de attribuutsoort moet aan één van deze datatypen voldoen.

*Voorbeeld: Attribuutsoort geometrie met als type de Union PuntOfVlak. PuntOfVlak is daarbij een Union met union element: punt, met als type het datatype GM_Point en union element vlak met als type het datatype GM_Surface.*

In dit voorbeeld is er enkel een keuze tussen verschillende union elementen die zelf geen betekenisvolle context geven aan het te kiezen datatype. Er wordt in dit geval geen definitie gespecificeerd bij het union element (de definitie is optioneel).

In onderstaande voorbeelden is er wel sprake van een keuze tussen union elementen die een betekenisvolle context geven aan het te kiezen datatype. Er wordt in dit geval wel een definitie gespecificeerd bij het union element.

*Voorbeeld: Attribuutsoort hoogte met als type de Union BereikOfWaarde. BereikOfWaarde is daarbij een Union met union element 'bereik', met als type het datatype Interval en union element 'waarde' met als type het datatype Real.*

Regel: het is niet toegestaan dat union elementen binnen één en dezelfde union identiek zijn. De naam van elk union element moet verschillend zijn én de datatypen moeten op zijn minst verschillend zijn. Dus ofwel een ander datatype, ofwel hetzelfde datatype met een verschil in het metadata aspect patroon en/of formeel patroon hebben. Bijvoorbeeld: een keuze tussen een datatype CharacterString en een datatype CharacterString is alleen toegestaan als er een verschillend (formeel) patroon is gespecificeerd.

Merk op dat het mogelijk is om een eigen datatype te maken met een eigen naam en deze te gebruiken in een union element.

Wanneer een beoogd datatype uit een extern model komt en daar geen metamodel stereotype heeft, zoals bijvoorbeeld het geval is bij het GM package waarin een datatype als <<interface>> GM_Point is opgenomen, dan heeft dit datatype niet een KKG stereotype en mogelijk ook niet de UML-metaclass dataType. Het is dan aan de modelleur van het informatiemodel om te beoordelen of het type dan als datatype gebruikt kan worden. Het is niet gewenst om aan het externe model een stereotype toe te voegen, noch om in het externe model de UML-metaclass aan te passen.

## 3.5 Domeinwaarden of lijsten

In veel registraties wordt gewerkt met codetabellen om de mogelijke waarden van een attribuutsoort te specificeren. Deze mogelijke waarden kunnen op verschillende manieren worden opgenomen, afhankelijk van de gewenste stabiliteit:

**Enumeraties**.
Dit zijn statische lijsten, waaruit één waarde gekozen kan worden. De registratie en bijbehorende koppelvlakken kunnen erop vertrouwen dat er geen nieuwe waardes worden toegevoegd. Als er een nieuwe waarde bij komt wordt dit via een modelwijziging doorgevoerd. Dit wordt vooral toegepast bij lijsten die niet of weinig veranderen. Zo is bij het attribuutsoort Naamgebruik een enumeratie naamgebruik opgenomen met als waarden onder meer 'eigen' en 'eigen, partner'.

Als sprake is van dynamiek in de domeinwaarden wordt een **Referentielijst** of **Codelist** gebruikt. Dit betreft de situaties waarin domeinwaarden kunnen veranderen en/of het aantal domeinwaarden kan toe- of afnemen. De registratie en bijbehorende koppelvlakken worden dan ingericht om hier mee om te gaan. Dit wordt vooral toegepast bij lijsten die vaker aan verandering onderhevig zijn.

**Referentielijst.**
Een lijst waarin we de betekenis en structuur van de lijst expliciet willen specificeren. Een voorbeeld is de referentielijst LAND of CultuurcodeOnbebouwd
( http://www.kadaster.nl/schemas/waardelijsten/CultuurcodeOnbebouwd ).

De referentielijst is hiermee een bijzondere vorm van datatype.
De naamgeving Referentielijst kan verwarring oproepen maar in principe wordt altijd gerefereerd naar gegevens m.b.t. één rij uit de referentielijst. In het geval van de referentielijst LAND wordt altijd gerefereerd naar gegevens over Nederland (NL) of gegevens over Duitsland.

Let op, wanneer er voor een bepaald attribuut in een informatiemodel of koppelvlak van een andere organisatie gekozen is voor een referentielijst, en uw organisatie koppelt hiermee, dan is het (meestal) onverstandig om in het eigen informatiemodel dit te behandelen als enumeratie.

Modelleerrichtlijn: elk attribuut van een object heeft een specifieke lijst met toegestane waarden. Modelleer daarom elke waardelijst bij voorkeur specifiek, met een eigen naam en locatie. Het is mogelijk dat de

structuur van de data gelijk is voor een aantal lijsten. Er kan dan een generieke structuur gemodelleerd worden, bv. een abstracte referentielijst met naam *_Waardelijst,* waarvan de specifieke waardelijsten overerven. Het metagegeven locatie is immers specifiek voor één waardelijst en moet per individuele waardelijst vastgelegd worden.

**CodeList**
Gebruik een codelist als in het informatiemodel de attributen, zoals bij een referentielijst, niet relevant zijn en je voor de definitie alleen wilt verwijzen naar de externe waardelijst.

**Waardenbereik en patroon**
In plaats van een opsomming van alle mogelijke waarden via een enumeratie of referentielijst leggen we soms voor een attribuut (attribuutsoort, referentie element, data element of union element) alleen het waardenbereik vast door middel van een patroon. Bijvoorbeeld bij een postcode, of een gemeentecode die moet bestaan uit precies 4 getallen, waarbij het eerste getal een 0 mag zijn.
Er wordt met een patroon expliciet niet een Enumeratie bedoeld. Het betreft een andersoortige, meer open beschrijving van welke waarden toegestaan zijn voor een attribuutsoort.

We onderkennen twee soorten patronen:
- *Patroon*: het metagegeven (de tagged value) 'Patroon' in tekst vorm. Deze wordt als aanvulling op het datatype (bijvoorbeeld Integer ) van het attribuut gespecificeerd. Het patroon bevat een specificatie waaraan een waarde moet voldoen. Bijvoorbeeld een postcode, met als aanduiding van het patroon: Postcode. De toegestane waarden voor deze patroon aanduiding worden dan vastgelegd in documentatie behorende bij het type: alle postcodes van 1000AA tot en met 9999ZZ.
- Formeel patroon: het metagegeven (de tagged value) 'Formeel patroon' in formele specificatie vorm [H1.11, referentie 6], te weten in een reguliere expressie. Bijvoorbeeld een postcode, met de expressie: \d{4}[A-Z]{2}

Een voorbeeld waar een patroon nodig is, is een attribuut waarvan het waardenbereik altijd een getal is met precies de lengte 4, zoals bijvoorbeeld 0001 tot en met 9999, en dus voorloopnullen heeft. Een datatype zoals Integer kan hiervoor niet gebruikt worden, omdat 0001 geen getal is. Het type van het attribuut wordt in dat geval een CharacterString, met lengte (exact) =4 en het patroon voor het attribuut specificeert dat alleen numerieke getallen zijn toegestaan: [0-9]{4}.

## 3.6 Abstracte objecttypes en concrete objecten

Een objecttype kan aangeduid worden als een abstract objecttype (zie paragraaf 2.3.1. door middel van indAbstract = J). Het betreft dan altijd een generalisatie waarbij de specialisaties van dit objecttype op het laagste niveau concrete objecttypen worden genoemd. Het is belangrijk te weten wanneer een objecttype als abstract objecttype in een informatiemodel is te onderkennen. Omdat een abstract objecttype altijd een generalisatie is, beantwoorden we in deze paragraaf ook de vraag wanneer we specialisaties / generalisaties onderkennen in een conceptueel informatiemodel en in een logisch informatiemodel

**Conceptueel informatiemodel**

*Specialisatie / generalisatie*
Bovenstaande vragen beantwoorden we aan de hand van een voorbeeld: het opleidingsinstituut.
In de beschouwde werkelijkheid onderscheiden we onder meer als gespreksonderwerp personen. Deze personen kunnen docenten en leerlingen zijn. Over al deze gespreksonderwerpen willen we gegevens communiceren. Een docent heeft als kenmerk dat deze een arbeidscontract met het opleidingsinstituut heeft afgesloten en een lesbevoegdheid heeft, terwijl een leerling kenbaar heeft gemaakt lessen te willen gaan volgen bij het instituut en dus geen arbeidscontract heeft afgesloten. Docenten en leerlingen zijn personen die rechten en plichten hebben.

Docent is een specialisatie ('subtype') van het objecttype Persoon en Leerling is een specialisatie van het objecttype Persoon. Een specialisatie ontstaat derhalve doordat aan een object van een bepaald type speciale eisen wordt gesteld. Vice versa spreken we er van dat Persoon een generalisatie is van Docent en Leerling. Op onderdelen vertonen de onderscheiden objecttypen Docent en Leerling hetzelfde gedrag waarbij dat gedrag essentieel van belang is voor het te beschouwen domein en daarmee het conceptuele informatiemodel.

*Abstract / concreet*
Wanneer er vanuit wordt gegaan dat binnen het te beschouwen gebied een persoon altijd ofwel een docent ofwel leerling kan zijn (en nooit beide tegelijk) dan definiëren we een Persoon als een abstract objecttype. Docent en Leerling zijn dan concrete objecttypen in het conceptueel informatiemodel.

Een concreet object kan zich alleen in de hoedanigheid als één van de specialisaties van het abstracte objecttype op het laagste niveau voordoen. En daarmee dus ook in de hoedanigheid van de generalisatie(s) van het concrete objecttype.

Richtlijnen
- Een abstract object is een onderwerp van gesprek binnen het beschouwde gebied. Het heeft dus echt een betekenis[15] in het beschouwde gebied. Net als een concreet object, specialisatie of generalisatie.
- De definitie van elk objecttype (dus ook een abstract objecttype) is zodanig dat ondubbelzinnig bepaald kan worden dat een object wel of niet tot het gedefinieerde type behoort. Dus niet een objecttype met een definitie als 'De gemeenschappelijke eigenschappen van een object…' Deze definitie is op elk objecttype van toepassing.
- Houd er rekening mee dat afhankelijk van het te beschouwen gebied een object uit de werkelijkheid in het ene informatiemodel een concreet objecttype kan zijn en in het andere informatiemodel een abstract objecttype.

**Logisch informatiemodel**
In een logisch model gelden in principe dezelfde regels voor abstracte of concrete objecttypen. De abstracte objecten worden daarom in principe overgenomen. In een logisch model wordt het digitale model van de werkelijkheid beschreven. Vanuit dit oogpunt kunnen er ook andere redenen zijn om abstracte objecttypen te creëren, bijvoorbeeld omdat een abstract object positieve effecten heeft voor de implementatie. Denk hierbij aan het aanbrengen van (extra) hiërarchie,zowel in semantiek als in ordening van eigenschappen. De enige regel die geldt is dat een abstract objecttype niet geïnstantieerd kan worden. Elk object is altijd een instantie van een concreet objecttype.

*Algemeen*

- In UML wordt een abstract objecttype aangeduid door indAbstract met waarde "J" en wordt de naam van het abstracte objecttype cursief geschreven (voorbeeld: *Persoon*).
- Een objecttype dat een generalisatie-relatie heeft naar een al dan niet abstract objecttype noemen we een specialisatie (van dat objecttype). Wanneer de generalisatie een abstract objecttype is, kan de specialisatie zelf ook abstract zijn en specialisaties hebben. De 'onderste' specialisaties zijn dan altijd concreet (niet abstract)
- Modelleer een abstract objecttype pas wanneer er sprake is van twee of meer specialisaties
- De unieke aanduiding kan opgenomen worden in het abstracte objecttype. De unieke aanduiding geldt dan voor elke specialisatie van het abstract objecttype. Dit betekent dat de unieke aanduiding uniek is binnen de verzameling van alle objecten die als specialisatie onder het abstracte objecttype vallen. Anders gezegd, de unieke aanduiding geldt voor alle concrete objecten die als objecttype de unieke aanduiding erven van het abstracte objecttype. Bijvoorbeeld: als de unieke aanduiding van *_Kadastraal object* het attribuutsoort Identificatie is en *_Kadastraal Object* als specialisatie een Perceel kent en een Leidingnetwerk, dan kan het niet zo zijn dat er een perceel object is met identificatie '1' en een leidingnetwerk met identificatie '1'. Als dat wel het geval is, dan moet op beide concrete objecttypes een eigen unieke aanduiding gedefinieerd worden.

## 3.7 Relatieklasse (uitzonderingen)

De gegevens van de relatiesoort worden altijd voor één relatiesoort vastgelegd. Het is echter mogelijk dat dezelfde gegevens voor meerdere relaties tegelijk gelden. Het is dan niet mogelijk om het te modelleren als relatieklasse. Wel gewenst, maar het kan niet als UML-associationClass. Als deze uitzondering het geval is, dan worden de relatieklasses gemodelleerd als «Objecttype», met één inkomende relatie en één uitgaande relatie. De oorspronkelijke kardinaliteit van de beoogde relatieklasse wordt hierbij behouden.

*Een Perceel kan vanwege een Perceel splitsing overgaan in twee of meerdere andere Percelen. De 'overgegaan in' relatie wordt bijgehouden in een relatieklasse. Gegevens over de splitsing zijn voor al deze relaties gelijk.*

Het metamodel ondersteunt (nog) geen relatieklassen tussen drie of meer objecttypen. Dit kan in uw eigen extensie toegevoegd worden.

*Bijvoorbeeld: een CONTRACT kan bijvoorbeeld ook een afspraak zijn tussen twee óf méér SUBJECTen, waarbij de gegevens van de relatie voor alle betrokken objecten hetzelfde zijn. CONTRACT wordt dan gemodelleerd als objecttype, waarbij beschreven wordt wat er moet gebeuren wanneer één van de SUBJECTen niet meer bestaat.*

## 3.8 Constraint

Deze paragraaf gaat dieper in op hoe een Constraint toegepast wordt.

Een Constraint wordt beschreven met een:

- Naam (UML-constraint name): een naam c.q. label, vaak in steekwoorden.
- Specificatie in tekst (UML-Constraint Notes, type invariant): een uitgebreide heldere beschrijving van de constraint in gewone tekst.

en optioneel:

- Specificatie formeel (UML-Constraint Notes, type OCL): formele specificatie in de Object Constraint Language. De formele specificatie bevat dus de uitgebreide heldere beschrijving van de constraint in gewone tekst EN de formele specificatie in OCL.

Twee constraints die gedefinieerd zijn op hetzelfde modelelement mogen niet dezelfde naam hebben.

In Enterprise architect 12.x lijkt het helaas (nog) niet mogelijk om constraints zoals bedoeld in UML op te nemen, te weten als OpaqueExpression met een constraint string en een aanduiding van de taal: natuurlijke taal, of OCL (of een andere zoals Java, maar deze wordt niet toegepast in dit metamodel). EA werkt met UML Notes en een constraint type. Het is daarnaast niet mogelijk om de tekst en de OCL in dezelfde constraint op te nemen. Het worden dan twee aparte constraints, 1 met tekst en 1 met OCL, met verplicht ook een andere naam. Vandaar onderstaande aanpak:

Als de modelleur kiest om de constraint alleen in gewone taal te beschrijven, dan als volgt:

- Naam (UML-constraint name): een naam c.q. label, vaak in steekwoorden.
- Specificatie in tekst (UML-Constraint Notes, type invariant): een uitgebreide heldere beschrijving van de constraint in gewone tekst.

Als de modelleur kiest om de constraint niet alleen in gewone taal te beschrijven, maar ook in een formele taal (OCL), dan als volgt:

- Naam (UML-constraint name): een naam c.q. label, vaak in steekwoorden.
- Specificatie formeel (UML-Constraint Notes, type OCL): formele specificatie in de object constraint language (OCL). De uitgebreide heldere beschrijving van de constraint in gewone tekst wordt opgenomen als commentaar, tussen /* */.

Aanbeveling: als een eigenschap van één UML-attribute, of één UML-association met een patroon (zie patroon) of een lengte (zie metadata aspect) of een kardinaliteit van een relatiesoort vastgelegd kan worden, gebruik dan deze en geen UML-constraint. Als er sprake is van een eigenschap die over meerdere informatiemodelelementen heen gaat, dan is er altijd sprake van een regel die we modelleren met een UML-constraint.

Aanbeveling: wees terughoudend met het gebruik van constraints in het informatiemodel wanneer de kans reëel is dat het model hierdoor gaat wijzigen of als het niet direct over de semantiek, structuur of syntax van de te registreren gegevens gaat. Dit is bijvoorbeeld het geval wanneer er regels bestaan rondom informatie die elke paar jaar kan wijzigen, of die per toepassingsgebied (net) anders toegepast moet worden. Bijvoorbeeld: wanneer een persoon 65 jaar of ouder is, mag deze geen uitkering aanvragen. Wanneer er veel van zulke constraints in het informatiemodel worden opgenomen, zal dit leiden tot een ongewenste dynamiek waardoor er (te) vaak nieuwe versies moeten worden uitgebracht. De aanbeveling is om de specificatie van dergelijke constraints buiten het informatiemodel te specificeren, bijvoorbeeld als validatieregel.

## 3.9 Historie

Deze paragraaf geeft in meer detail aan wat we onder de metagegevens *Indicatie materiële historie en Indicatie formele historie* verstaan.

Aanvullend beschrijft deze paragaaf een aantal aspecten waar rekening mee gehouden kan worden bij de uitwerking van historie in een informatiemodel op logisch niveau. Let wel, historie is niet gestandaardiseerd over logische informatiemodellen van registraties heen. Hoe historie in een logisch informatiemodel wordt gemodelleerd is aan de opsteller van het logisch informatiemodel zelf. In deze paragraaf wordt wel iets verteld over historie op logisch niveau maar dat is niet bindend!

**Algemeen**

Het aspect tijd speelt een belangrijke rol in (basis)registraties. Daarnaast speelt tijd ook een belangrijke rol in het gebruik van de informatie uit (basis)registraties. Afnemers hebben eigen rechtsprocedures en moeten kunnen herleiden wanneer een gegeven als bekend mocht worden verondersteld. Als bijvoorbeeld besluiten ter discussie worden gesteld, is het juridisch van belang te achterhalen op basis van welke gegevens zo'n besluit is genomen. Als onjuiste gegevens zijn gebruikt, is het relevant te weten of het juiste gegeven tijdens de besluitvorming al bekend waren.
Dit betekent dus dat bekend moet zijn wat de waarde van een attribuut op een bepaald moment is.

*Tijdslijnen*

Er spelen twee tijdslijnen een rol bij het herleiden van attribuutwaarden:

- Wanneer is iets gebeurd, in de werkelijkheid of volgens opgave (wanneer zijn de opgenomen gegevens geldig)? Dit valt binnen de tijdlijn van de aangehouden werkelijkheid.
- Vanaf wanneer wist de overheid (als collectief van organisaties) dat de gegevens bekend waren? Dit valt binnen de tijdlijn van het administratieproces of de administratieve werkelijkheid.

In de rapportage 'Architectuur van het stelsel' (Stroomlijning BasisGegevens, 2006) wordt geadviseerd om beide tijdslijnen te registreren, om de attribuutwaarden van een bepaald moment te kunnen reconstrueren. In de diverse registraties wordt hieraan op verschillende wijzen invulling gegeven. Dit metamodel schrijft derhalve niet voor welke bij de tijdslijnen behorende attributen gebruikt moeten worden voor het vastleggen van historie.

**Historie op conceptueel niveau**

Op conceptueel niveau is het wel altijd mogelijk om aan te geven dát het bijhouden van historie *aan de orde is* voor een (elk) gegeven, dat wil zeggen een attribuut of relatie van een object, te weten via een metagegeven. Deze metagegevens specificeren we als volgt:

- *Indicatie materiële historie*: indicatie of de materiële historie van de attribuutsoort te bevragen is. Materiële historie geeft aan wanneer een verandering is opgetreden in de werkelijkheid die heeft geleid tot verandering van de attribuutwaarde. Materiële historie impliceert dat actuele, historische en eventuele toekomstige attribuutwaarden te bevragen zijn.
- *Indicatie formele historie*: indicatie of de formele historie van de attribuutsoort te bevragen is. Formele historie geeft aan wanneer in de administratie een verandering is verwerkt van de attribuutwaarde (wanneer was de verandering bekend en is deze verwerkt).

*Voorbeelden: 'bouwjaar pand' heeft al materiële historie in zich: het bouwjaar is het moment waarop de wijziging in de werkelijkheid zich voordeed en wijzigt niet. De 'indicatie materiële historie' ervan is daarom Nee. BSN van een Persoon geldt voor de persoon vanaf het moment dat de persoon in de BRP is opgenomen en wijzigt niet: 'indicatie materiële historie' Nee. De Achternaam van een persoon kan*

*wijzigen: 'indicatie materiële historie' Ja. Als je niet toeziet op het daadwerkelijk kappen van een boom maar het gekapt zijn wel in een registratie wil opnemen: 'indicatie materiële historie' Nee en 'indicatie formele historie' Ja.*

Richtlijn: op conceptueel niveau worden voor historie alléén indicatie materiële historie en indicatie formele historie bij een attribuut of relatie vastgelegd, en dus géén bij de tijdslijnen behorende attributen die gebruikt moeten worden voor het vastleggen van historie. Deze bij de tijdslijn behorende attributen worden op het logische niveau vastgelegd.

**Historie op logisch niveau**

KKG schrijft[16] geen implementatie van het conceptuele niveau voor. Wel worden er aandachtspunten gegeven om rekening mee te houden. Denk bij de uitwerking o.a. aan de volgende aspecten:

- Het bijhouden van historie met specifieke attributen per objecttype, zoals bijvoorbeeld: bouwjaar pand, of met generieke tijdslijnattributen attributen die gelden voor alle objecttypes, zoals begindatum geldigheid. Denk aan de attribuutsoort en/of gegevensgroeptype (herbruikbaar);
- historie bijhouden per attribuut (en relatie) of per versie van een object. Bij deze laatste kan de gegevensgroep gekoppeld worden aan bijvoorbeeld elk objecttype;
- de status transities die een object in zijn levenscyclus doorloopt. Er kan ook gekozen worden om deze in een conceptueel informatiemodel op te nemen, als ze op dat niveau al van belang zijn;
- status attributen, die iets zeggen *over* gegevens, zoals attributen die aangeven wel of niet beschouwd moet worden als onderdeel van de geldige gegevens van een object. Er kan ook gekozen worden om deze in een conceptueel informatiemodel op te nemen, als ze op dat niveau al van belang zijn. Voorbeeld: BAG inactief. Het kan ook van belang zijn om aan te geven dat een gegeven niet terug te vinden is in een brondocument (omdat er fouten zijn gemaakt bij de verwerking), maar niet verwijderd kan worden omdat alle gegevens, ook foutieve, bestendig moeten worden bewaard. Voorbeeld: BAG indicatieNietBAG.

Aanbeveling: het komt voor dat er binnen één domein, van één conceptueel informatiemodel, meerdere logische informatiemodellen worden uitgewerkt. Het is dan aan te beleven om centrale implementatie afspraken op te stellen, welke gelden voor elk logisch informatiemodel. Dit speelt met name rondom historie, omdat het vaak ongewenst (en erg Gestructureerdof zelfs onmogelijk) is om verschillende implementaties naast elkaar in stand te houden en naar elkaar te vertalen.

Opmerking: de metagegevens Indicatie materiële historie en Indicatie formele mogen worden opgenomen in een logisch model (of worden overgenomen van het conceptuele naar het logische informatiemodel).

[16] Hoewel het goed zou zijn om tot een standaard te komen in Nederland is KKG niet de plek hiervoor.

## 3.10 Afleidbare gegevens

In een informatiemodel kan de behoefte bestaan om afgeleide gegevens op te nemen: dit zijn gegevens die afleidbaar zijn uit andere attribuut- en/of relatiesoorten binnen het informatiemodel. Dit lijkt op redundantie. Toch hebben we deze gegevens daar opgenomen waar er ten eerste vraag is naar het afgeleide gegeven en ten tweede het gegeven niet eenvoudig af te leiden is (er moet sprake zijn van enige mate van complexiteit). Dit wordt in UML weergegeven via isDerived. Zie ook Attribuutsoort, §2.4.2 – is afleidbaar.

*Voorbeeld is de 'Datum vestiging in Nederland' van een Ingeschreven persoon. De afleiding van dit gegeven is niet triviaal. Door het als afleidbaar gegeven op te nemen kan het opgevraagd worden zonder dat de historie of andere gegevens van het object opgevraagd hoeven te worden om daaruit dit gegeven af te leiden.*

## 3.11 Authentieke gegevens

Bij een attribuutsoort of relatiesoort wordt als metagegeven 'Authentiek' opgenomen. Het is een aanduiding of een attribuutsoort of een als relatiesoort gemodelleerd landelijk basisgegeven in de catalogus van de desbetreffende registratie een authentiek gegeven betreft. Een authentiek gegeven is van hoogwaardige kwaliteit en kan zonder nader onderzoek bij de uitvoering van publiekrechtelijke taken worden gebruikt.

De specificatie van de waarde van het metagegeven is gebaseerd op het onderscheid in de volgende groepen van gegevens:

- Landelijke registraties met authentieke en niet-authentieke basisgegevens (BAG, BRK, BRP, BGT e.d.);
- Landelijke sector- en domein-overstijgende informatiemodellen (IMGeo e.d.);
- Gemeentelijke sector- en domein-overstijgende informatiemodellen (RSGB, RGBZ, ZTC);
- Sector- en domein-specifieke informatiemodellen (LV-WOZ, IMRO e.d.).

| **Waardebereik authentiek** | **Betekenis** |
|---|---|
| Authentiek | Indien het een authentiek (landelijk) basisgegeven of een als relatiesoort gemodelleerd authentiek (landelijk) basisgegeven is. Basisgegevens zijn altijd gegevens afkomstig uit de landelijke registraties. |
| Basisgegeven | Indien het een landelijk basisgegeven of een als relatiesoort gemodelleerd (landelijk) basisgegeven is in een landelijke registratie, maar in die registratie géén authentiek gegeven is. |
| Landelijk kerngegeven | Indien het een gegeven of een als relatiesoort gemodelleerd gegeven is in een landelijk sector- en domein-overstijgend informatiemodel en geen authentiek gegeven en geen basisgegeven is. |
| | |
| Overig | Indien het géén van de voorgaande categorieën betreft. Veelal gaat het dan om proces-, taakveld- of domeinspecifieke gegevens. |

## 3.12 Mogelijk geen waarde

Een attribuut kan geen waarde hebben, omdat de waarde optioneel is en er niet is. Bijvoorbeeld bij een tussenvoegsel van een achternaam. Maar een attribuut kan ook mogelijk geen waarde hebben, omdat de waarde niet bekend is. Dat er geen waarde bij een attribuut geregistreerd is, wil dus niet zeggen dat er geen betekenis aan gehecht kan worden. Zo kan het niet hebben van een waarde van de overlijdensdatum van een persoon betekenen dat deze persoon nog leeft. Maar het kan ook betekenen dat de persoon overleden is maar de datum waarop deze persoon overleden is, niet bekend is.

Dit verschil is niet vast te leggen zonder onderscheid te maken en vaak is het ook van belang om de reden waarom de waarde niet bekend is vast te leggen. Een verplicht veld optioneel maken is daarom niet de juiste oplossing. In die situaties waarin het hebben van geen waarde van een attribuut een betekenis kan hebben maken we gebruik van het metagegeven 'Mogelijk geen waarde'. Dit metagegeven geeft op informatiemodelniveau aan dat het attribuut een gangbare waarde kan hebben, maar dat deze waarde ook niet bekend kan zijn.

Bij de daadwerkelijke registratie kan het zo zijn dat:

- De waarde van het attribuut bekend is, te weten een waarde bij een verplicht attribuut, of geen waarde bij een optioneel attribuut.
- De waarde van het attribuut onbekend is, en niet meer kan worden achterhaald.
- De waarde van het attribuut onbekend is, en mogelijk wel nog kan worden achterhaald.

Wat de toegestane redenen zijn voor een specifiek attribuut, is aan de beheerder van het informatiemodel. Het is nuttig om de redenen te beperken op informatiemodelniveau. Dit kan dan vastgelegd worden bij de attribuutsoort of bij relatiesoort zelf, bijvoorbeeld in de UML notes. In de registratie mogen dan alleen deze redenen worden geregistreerd.

Een attribuut dat in de werkelijkheid gewoon geen waarde kan hebben en waar bovenstaand onderscheid niet van toepassing is duiden we niet aan met dit metagegeven. Het betreft dan gewoon een optioneel attribuut dat niet is gevuld. Anders gezegd, het is bekend dat het attribuut niet gevuld is en het hebben van geen waarde heeft dan geen verdere betekenis .

Ook een relatiesoort of compositie relatie kan mogelijk geen waarde hebben waaraan betekenis gehecht kan worden en ook daar maken we gebruik van het metagegeven 'Mogelijk geen waarde'.

In de registraties komen we hier en daar enumeraties tegen waarin de waarde 'onbekend' is opgenomen. Bijvoorbeeld de geslachtsaanduiding van een natuurlijk persoon. De enumeratie bestaat uit de waarden man, vrouw en onbekend. In dit metamodel stellen we dat dit niet mag c.q. niet de bedoeling is bij de modellering van eigen gegevens in een eigen informatiemodel. Uitzondering is wanneer het een situatie betreft waarin gegevens worden overgenomen uit een registratie die wel de waarde 'onbekend' gebruikt. Dan kan er ook gekozen worden voor het 1:1 overgenomen van de gegevensdefinitie uit deze andere registratie.

## 3.13 Externe schema's (her) gebruiken

In bepaalde situaties is het mogelijk dat een ander informatiemodel al één op één de specificaties in UML bevat die relevant zijn voor het eigen informatiemodel. Dit is in het bijzonder het geval als het andere informatiemodel ook dit metamodel volgt, maar kan ook het geval zijn bij gestandaardiseerde datatypes. Het is dan wenselijk om hiernaar te kunnen verwijzen. Dit kan door deze packages over te nemen naar de eigen UML tool en het stereotype «extern» toe te kennen. Deze packages worden dan wel buiten het eigen

informatiemodel gehouden. Ze zijn extern aan het eigen model. Het beheer en de definitie vindt dan ook buiten het eigen model plaats.

In deze externe packages die aangeduid worden met het stereotype «extern» zijn de relevante specificaties opgenomen die binnen het informatiemodel hergebruikt worden. Deze specificaties zijn opgesteld door een externe partij die de UML (of ook de XML) schema's beheert en beschikbaar stelt waarnaar vanuit deze specificaties wordt gerefereerd. De packages bevatten alleen de constructies die ook daadwerkelijk binnen het 'eigen' informatiemodel wordt gebruikt.

*Voorbeeld: voor het uitwisselen van geografische informatie op basis van NEN3610 is een tweetal externe packages onderkend waarnaar vanuit de 'eigen' informatiemodellen kan worden verwezen: NEN3610, of GML3.2*

Het is ook mogelijk om binnen een domein of binnen een organisatie een eigen «extern» package te definiëren voor datatypen, om over meerdere informatiemodellen heen hergebruik mogelijk te maken.

Naast het beschikbaar maken van het externe package kan het modelelement uit het externe package gebruikt worden als datatype, maar er kan ook naar verwezen worden via een relatie. Dit laatst wordt nader uitgelegd in de volgende paragraaf.

## 3.14 Koppelen met ander informatiemodel (externe koppeling)

Bij registraties  is het regelmatig noodzakelijk om te verwijzen vanuit het eigen model naar gegevens uit een andere informatiemodel. Denk aan het opnemen van de identificatie van een object uit een andere registratie, of aan het overnemen van een subset van gegevens van een object uit een ander model. Hiervoor zijn de stereotypes «view» en «externe koppeling» bedoeld.

Deze stereotypes zijn alleen van toepassing binnen een informatiemodel in situaties waarbij het ene informatiemodel afhankelijk is van een andere informatiemodel.

Uitgangspunten hierbij zijn dat de definitie van de structuur van gegevens van het andere informatiemodel één op één overgenomen wordt, waarbij expliciet gemaakt wordt welke gegevens tot het eigen model behoren en welke tot het andere model.

Bijvoorbeeld:
*In IMKAD zit het objecttype: «Objecttype» Persoon. Hierin zitten de attributen waarvan de basisregistratie Kadaster de gegevens zelf inwint. In IMKAD zit het package: «view» BRP en hierin zit het «Objecttype» GeregistreerdPersoon. Hierin zitten de attributen die de basisregistratie BRP  inwint en die het Kadaster overneemt. De relatie overstijgt de registratie, máár het blijft in het eigen informatiemodel. De aard van de relatie is echter anders dan bij een «relatiesoort». Daarom kennen we deze relatie het stereotype «externe koppeling» toe.*

Het betreft in de werkelijkheid dezelfde persoon. Zowel Persoon als GeregistreerdPersoon worden als «Objecttype» gezien. Beide objecten zijn sterk aan elkaar gekoppeld. Dit is altijd zo bij dit soort relaties. Daarom is het aggregatietype van deze relatie altijd Composite.

Merk op dat er ook een relatie rechtstreeks naar de BRP gelegd had kunnen worden. Er is dan geen sprake van gegevens uit de BRP die overgenomen zijn in de eigen registratie. Er kan dan volstaan worden met alleen de unieke aanduiding van GeregistreerdPersoon. Dit is de BSN. Dit wordt niet gezien als een «externe koppeling» maar als een referentie.

## 3.15 Stelselcatalogus en stelselafspraken voor basisregistraties

Dit metamodel ondersteunt de metadata die voorgeschreven wordt voor de stelselcatalogus [H1.11, referentie 3]. Deze paragraaf geeft aan hoe de metadata in dit metamodel zich verhoudt tot die van de stelselcatalogus, zodat deze vanuit uw informatiemodel geleverd kunnen worden aan de stelselcatalogus. Er zijn ook stelselafspraken rondom metadata. Een metadata aspect in H2.4 met aanduiding √ is conform stelselafspraken voor basisregistraties. Beide gelden.

De metadata voor de stelselcatalogus en de metadata voor de stelselafspraken zijn beide verplicht voor basisregistraties. Als het informatiemodel géén basisregistratie is, kan je als organisatie zelf kiezen om (een aantal van) deze metadata buiten scope te plaatsen. Dit doe je in de eigen extensie, zoals beschreven in paragraf 1.8. De rest van deze paragraaf gaat alleen nog in op de metadata voor de stelselcatalogus.

Het metamodel gaat als volgt met de metadata van de stelselcatalogus om:

- Dit metamodel beschrijft de stelselcatalogus metadata alleen voor de metadata die op informatiemodel niveau speelt, niet de overige metadata.
- Dit metamodel neemt stelselcatalogus metadata altijd op met dezelfde semantiek/betekenis. Als de betekenis van metadata anders is, dan wordt ook niet dezelfde naam gebruikt. Als de betekenis gelijk is, dan kan het wel zo zijn dat dit metamodel een andere naam hanteert. De vertaling wordt hieronder weergegeven.
- Als de semantiek hetzelfde is, maar het waardenbereik van het gegeven is in dit metamodel een verdere verbijzondering (niet in strijd met) dan hanteert dit metamodel dezelfde metadata en geeft aan hoe de waarde in dit metamodel te vertalen naar de waarde in de stelselcatalogus. Bij automatische verwerking naar de stelselcatalogus is het wellicht dus soms nodig deze waardes om te zetten.

| Metadata in stelselcatalogus | Komt voor in Metamodel? | Waardenbereik hetzelfde? |
|---|---|---|
| *Naam* | J | J |
| *Definitie* | J | J |
| *Toelichting* | J | J |
| *Populatie* | J | J |
| *Herkomst* | J | J (met aanvullende afspraken) |
| *Authentiek* | J | N (met vertaling) |
| *Kwaliteit* | J | J |
| | | |
| *Relatie* | N | n.v.t. (kan wel via een vertaling) |
| *Wetgeving* | N | n.v.t. |
| *Eigenaar* | N | n.v.t. |
| *Toegankelijkheid* | N | n.v.t. |
| *Gebruiksvoorwaarden* | N | n.v.t. |

Waardenbereik afspraken

- *Authentiek:* als in dit metamodel 'Authentiek' (zie 2.4.21) dan 'Ja' in stelselcatalogus, anders Nee.
- *Herkomst:* Zelf in te vullen. Afspraken hierbij:
  Als zelf ingewonnen: noem de inwinnende organisatie. Bijvoorbeeld: KING of Gemeentes.
  Als overgenomen uit andere bron, noem de directe bron. Bijvoorbeeld: BAG.
- *Relatie:* dit is geen metagegeven in dit metamodel, maar een stereotype. Deze is wel af te leiden uit het metagegeven van relatiesoort: gerelateerd objecttype (de target van de relatie).

## 3.16 Naamgevingsconventies

Naamgevingsconventies zijn belangrijk om te specificeren. Onderstaande beschrijft enkele punten die op het niveau van dit metamodel zijn afgesproken. De verdere invulling van de naamgevingsconventies is aan de opsteller van het informatiemodel zelf (zie ook bijlage 1).

### 3.16.1 Alternatief 1: natuurlijke taal, die dichtbij de gebruiker staat

Met natuurlijke taal wordt bedoeld, zoals de gebruikers erover praten, in normaal Nederlands. Veelal zijn dit alleen letters en cijfers, met spaties. Koppeltekens (‘-’ of ‘_’) kunnen gebruikt worden, indien gewenst, alsmede diakrieten.

Zo kan bijvoorbeeld gekozen worden  dat de eerste letter een hoofdletter is voor namen van de modelelementen Objecttypen, Gegevensgroeptype en Datatypen en dat de eerste letter een kleine letter is voor attribuutsoorten,en data-elementen e.d. Bijvoorbeeld: ‘Natuurlijk persoon’ en ‘naam’ met type CharacterString.

Regel: voor conceptuele informatiemodellen wordt altijd alternatief 1 gehanteerd.

### 3.16.2 Alternatief 2: (ook) leesbaar door systemen

Met machine leesbare taal wordt bedoeld dat deze eenvoudig door systemen te verwerken is. Veelal zijn dit alleen letters en cijfers, zonder spaties, zonder diakrieten. Koppeltekens (‘-’ of ‘_’) kunnen gebruikt worden, maar dit wordt veelal vermeden.

Zo kan bijvoorbeeld gekozen worden voor UpperCamelCase voor namen van Objecttypen, Gegevensgroeptypen en Datatypen en lowerCamelCase voor attribuutsoorten, relatiesoorten, relatierollen, data-elementen e.d. Bijvoorbeeld: **n**aam in een NatuurlijkPersoon. Combineer indien nodig verschillende woorden (uit bijvoorbeeld een begrippenkader of uit een conceptueel informatiemodel) om precieze en begrijpelijke namen te vormen.

Als er gekozen wordt voor CamelCase, volg hierin dan hoe deze in UML ook toegepast wordt (deze komt overeen met ISO19103): maak van de beginletter van ieder deelwoord van namen van attribuutsoorten, relatierollen een hoofdletter, behalve de beginletter van het eerste woord. Bij namen van objecttypen, gegevensattributen, union, datatypen, en relaties wordt ook de beginletter een hoofdletter.

Regel: voor logische informatiemodellen wordt altijd alternatief 2 gehanteerd.

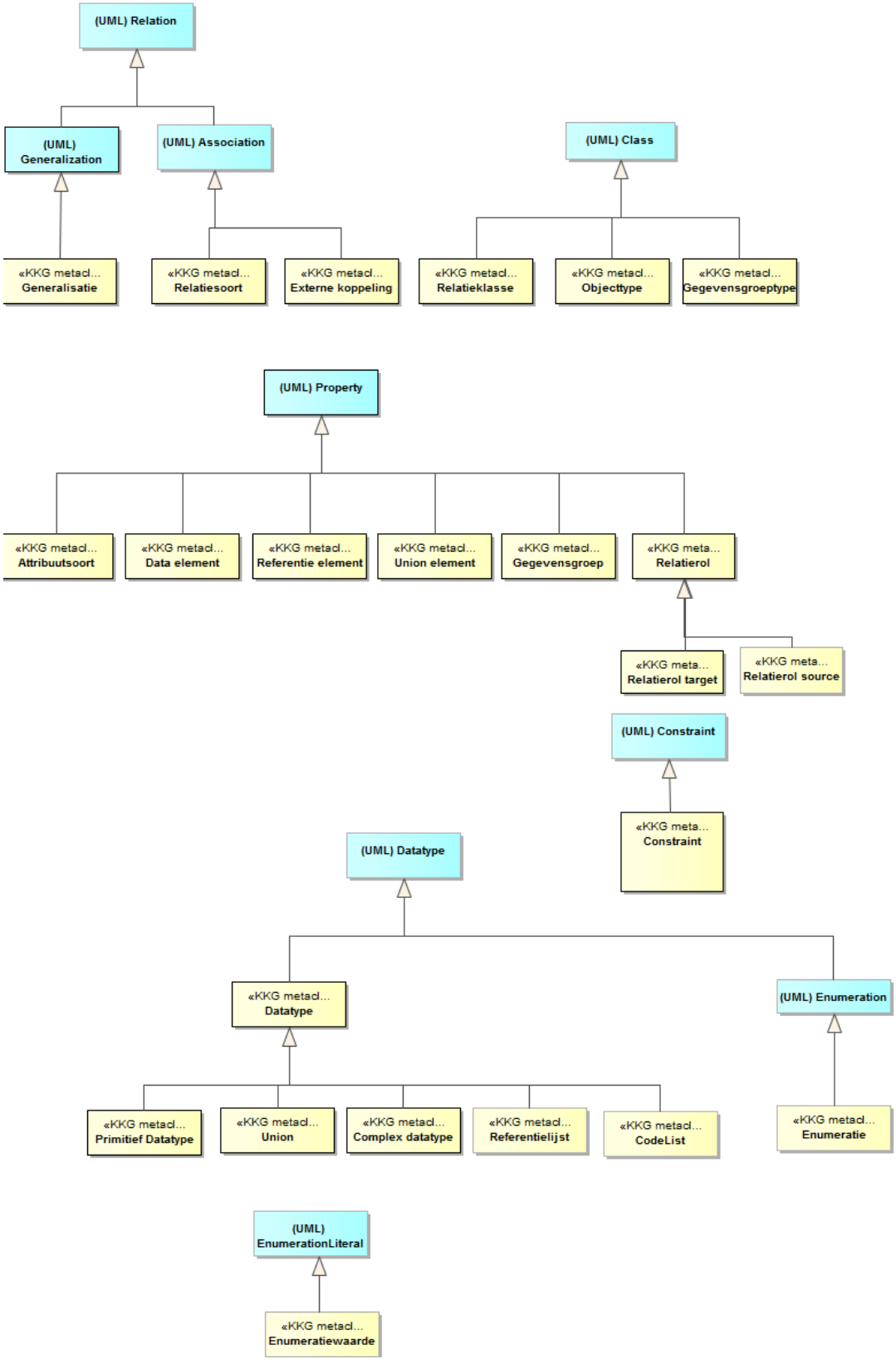
Neem aanvullend een verwijzing op naar het betreffende modelelement in het conceptuele model. Dit kan bijvoorbeeld met een trace of door opname van de naam in de alias (zie 3.16.20), zodat lezers goed de overgang van conceptueel naar logisch kunnen volgen.

### 3.16.3 Naamgeving voor metamodel elementen

Voor stereotypes en metagegevens worden dezelfde naamgevingsconventies toegepast als in alternatief 1 waarbij de eerste letter een hoofdletter is voor alle stereotypes en tagged values. Echter, als een internationale standaard het anders voorschrijft, dan wordt deze gevolgd, en niet vertaald. Bijvoorbeeld: codeList. Deze conventies gelden ook als in een eigen extensie metamodelelementen worden toegevoegd.

In de bijlage is een template opgenomen om de naamgevingsconventies in te specificeren. Dit is een hulptabel, die u over kunt nemen naar uw eigen extensie (zoals bedoeld in paragraaf 1.8) en in kunt vullen voor uw eigen informatiemodel (of organisatie).

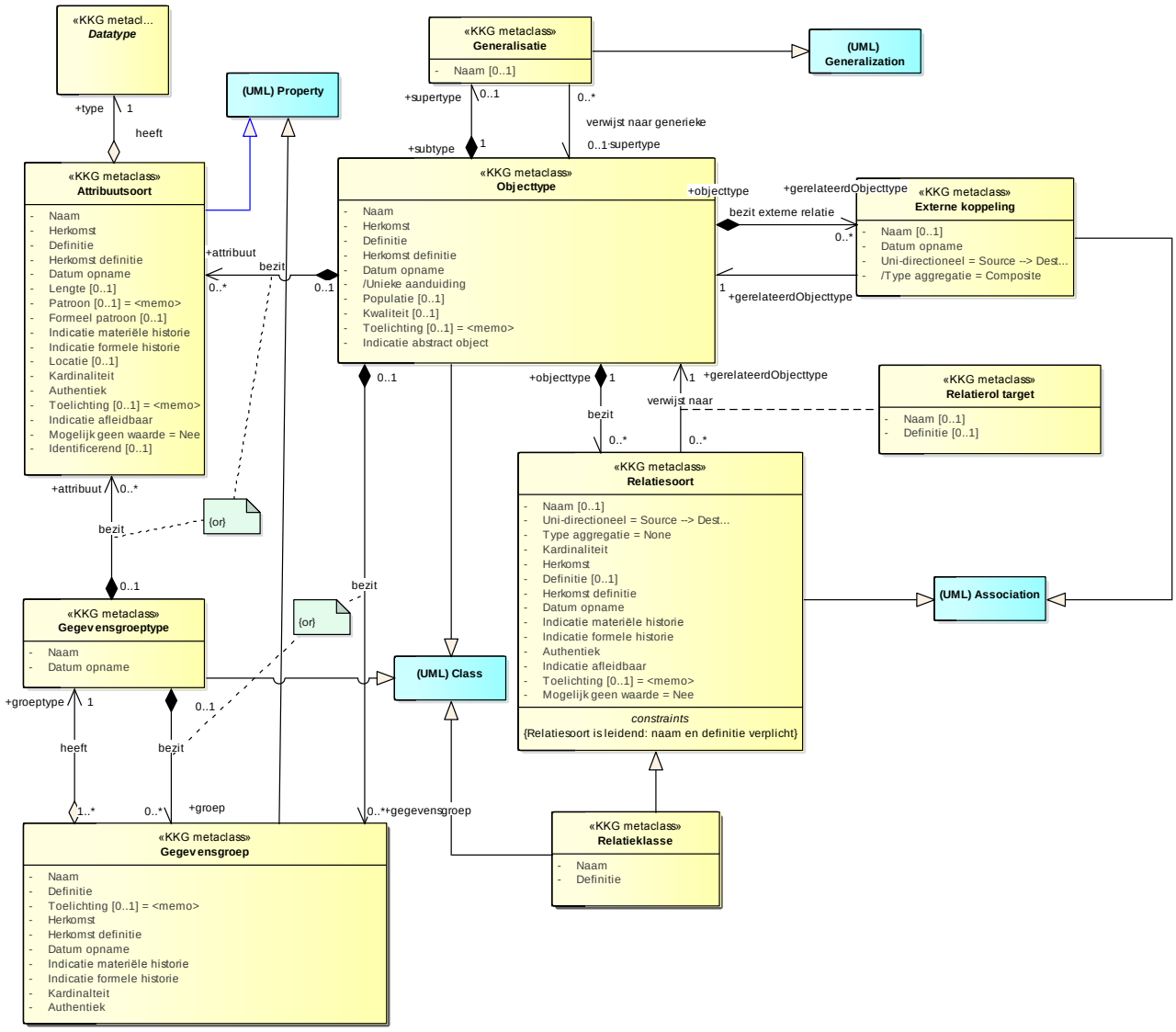
# Bijlage I: Overzicht toegepaste UML metaclasses



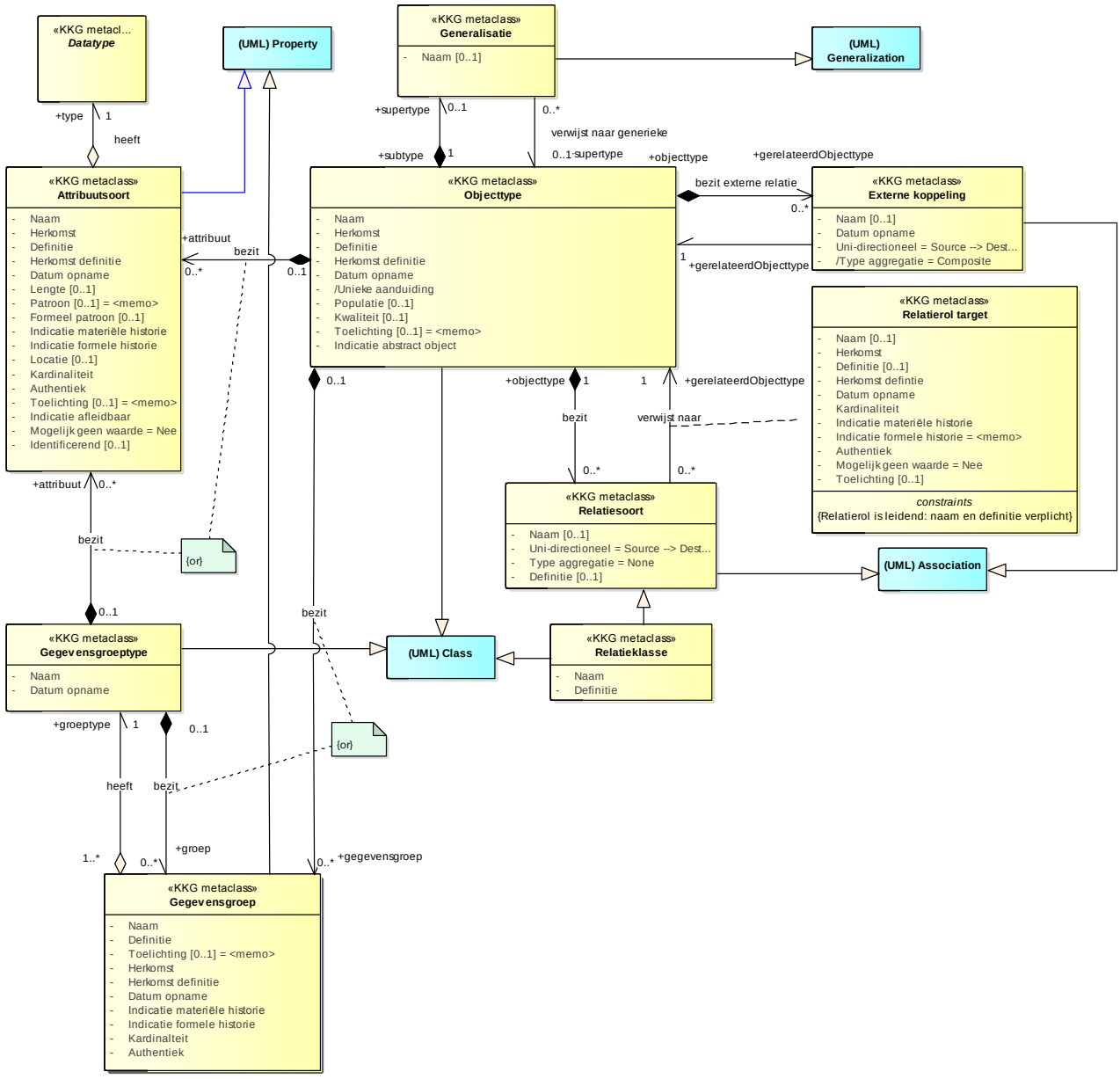
# Bijlage II: Modelelementen en metagegevens als diagram

Deze bijlage bevat alle modelelementen en metagegevens in één diagram.

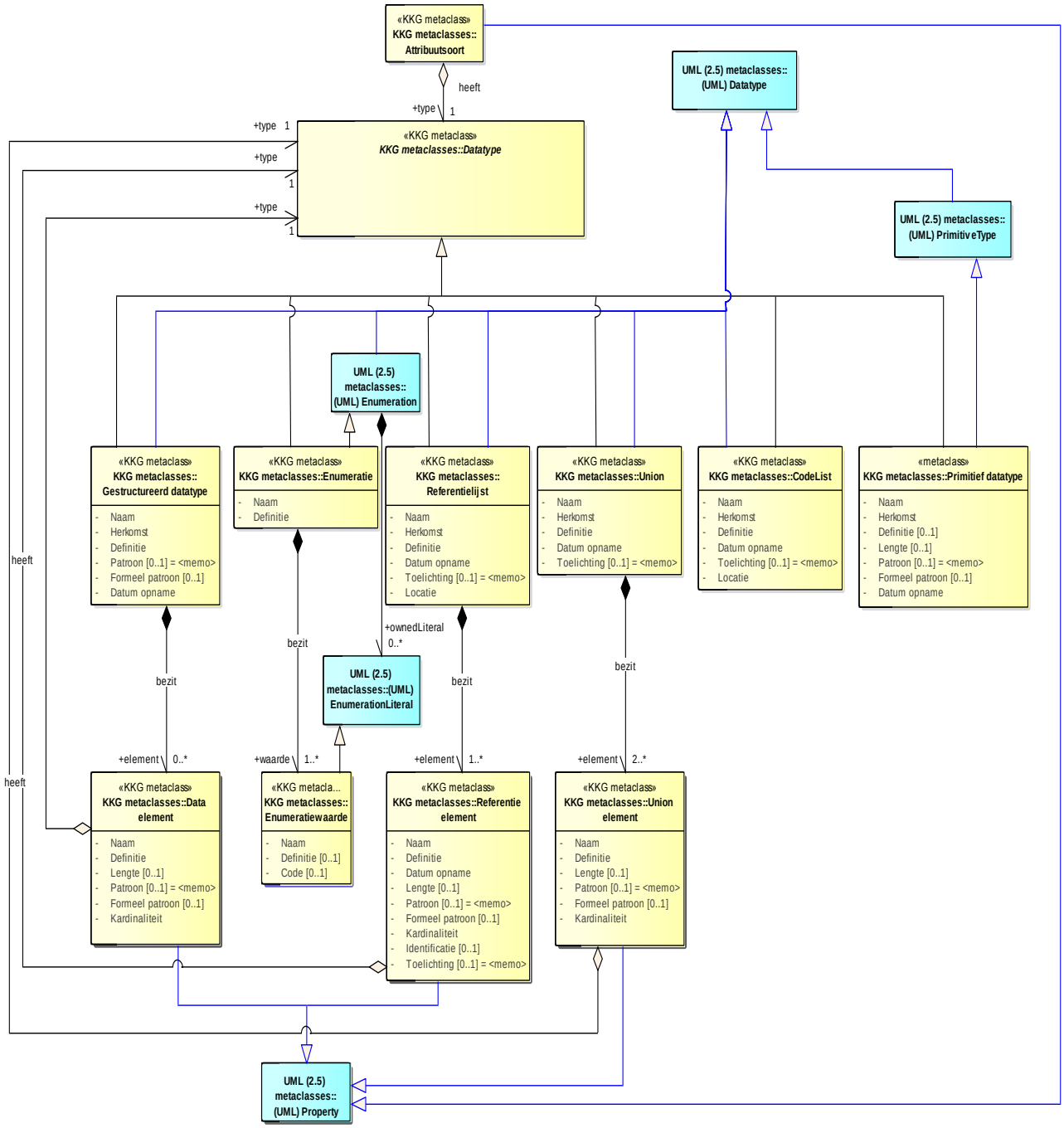
Kern - Relatiesoort is leidend (zie 3.2.3.1)



## Kern - Relatierol is leidend (zie 2.3.2.2)



## Datatypen


«KKG metaclass»
KKG metaclasses::
Attribuutsoort
heeft
+type
1
UML (2.5) metaclasses::
(UML) Datatype
+type 1
+type
1
+type
1
«KKG metaclass»
KKG metaclasses::Datatype
UML (2.5) metaclasses::
(UML) PrimitiveType
UML (2.5)
metaclasses::
(UML) Enumeration
«KKG metaclass»
KKG metaclasses::
Gestructureerd datatype
- Naam
- Herkomst
- Definitie
- Patroon [0..1] = <memo>
- Formeel patroon [0..1]
- Datum opname
«KKG metaclass»
KKG metaclasses::Enumeratie
- Naam
- Definitie
«KKG metaclass»
KKG metaclasses::
Referentielijst
- Naam
- Herkomst
- Definitie
- Datum opname
- Toelichting [0..1] = <memo>
- Locatie
«KKG metaclass»
KKG metaclasses::Union
- Naam
- Herkomst
- Definitie
- Datum opname
- Toelichting [0..1] = <memo>
«KKG metaclass»
KKG metaclasses::CodeList
- Naam
- Herkomst
- Definitie
- Datum opname
- Toelichting [0..1] = <memo>
- Locatie
«metaclass»
KKG metaclasses::Primitief datatype
- Naam
- Herkomst
- Definitie [0..1]
- Lengte [0..1]
- Patroon [0..1] = <memo>
- Formeel patroon [0..1]
- Datum opname
heeft
bezit
bezit
+ownedLiteral
0..*
bezit
bezit
UML (2.5)
metaclasses::(UML)
EnumerationLiteral
+element
0..*
+waarde
1..*
+element
1..*
+element
2..*
heeft
«KKG metaclass»
KKG metaclasses::Data
element
- Naam
- Definitie
- Lengte [0..1]
- Patroon [0..1] = <memo>
- Formeel patroon [0..1]
- Kardinaliteit
«KKG metacla...
KKG metaclasses::
Enumeratiewaarde
- Naam
- Definitie [0..1]
- Code [0..1]
«KKG metaclass»
KKG metaclasses::Referentie
element
- Naam
- Definitie
- Datum opname
- Lengte [0..1]
- Patroon [0..1] = <memo>
- Formeel patroon [0..1]
- Kardinaliteit
- Identificatie [0..1]
- Toelichting [0..1] = <memo>
«KKG metaclass»
KKG metaclasses::Union
element
- Naam
- Definitie
- Lengte [0..1]
- Patroon [0..1] = <memo>
- Formeel patroon [0..1]
- Kardinaliteit
UML (2.5)
metaclasses::
(UML) Property

## Bijlage III: Template naamgeving conventies

| **Modelelement** | **Naamgevingsconventie** | | **Voorbeeld** |
|---|---|---|---|
| | *Objecttype* | | |
| Naam objecttype | | | |
| | *Attribuutsoort* | | |
| Naam attribuutsoort | | | |
| | *Relatiesoort* | | |
| Naam relatie | | | |
| | *Gegevensgroeptype* | | |
| Naam gegevensgroeptype | | | |
| | *Gegevensgroep* | | |
| Naam gegevensgroep | | | |
| | *Externe koppeling* | | |
| Naam externe koppeling | | | |
| | *Relatieklasse* | | |
| Naam relatieklasse | | | |
| | *Referentielijst* | | |
| Naam referentielijst | | | |
| | *Referentie element* | | |
| Naam referentie element | | | |
| | *Gestructureerd datatype* | | |
| *Naam Gestructureerd datatype* | | | |
| | *Data element* | | |
| Naam data element | | | |
| | *Datatype* | | |
| *Naam datatype* | | | |
| | *Union* | | |
| *Naam Union* | | | |
| | *Union element* | | |
| *Naam union element* | | | |
| | *Enumeratie* | | |
| Naam enumeratie | | | |
| | *Enumeratiewaarde* | | |
| Code enumeratiewaarde | | | |
| Naam enumeratiewaarde | | | |