Geonovum Handreiking - Werkversie

WaU - Metadata basic principles

Geonovum Handreiking
Werkversie 15 juni 2023

Laatst gepubliceerde versie:
https://docs.geostandaarden.nl/dk/wau-metadata/

Redacteur:
Paul Janssen (Geonovum)

Auteur:
voornaam achternaam (Geonovum)

Doe mee:
GitHub Geonovum/NL-ReSpec-GN-template
Dien een melding in
Revisiehistorie
Pull requests

Dit document is ook beschikbaar in dit niet-normatieve formaat: pdf

Samenvatting

TODO: vul in abstract.md een abstract in.

Status van dit document

Dit is een werkversie die op elk moment kan worden gewijzigd, verwijderd of vervangen door andere documenten. Het is geen stabiel document.

Inhoudsopgave

§ 1. Introduction

Dit onderdeel is niet normatief.

Kadaster and Geonovum are developing a mechanism to facilitate the semantic integration of decentralized data registers. A central API operates on an integrated semantic layer. The semantic layer constitutes of a coherent information model integrating several domain specific domain models. An API orchestration layer directs data communication to and from existing data registries.

Data retrieval is according to the information model of the semantic layer. Data retrieval will optionally include metadata about the lineage (provenance) of the data.

This document specifies the lineage model used and the mechanism to relate the lineage data as metadata to published data.

§ 1.1 Scope

The scope of this lineage model is:

- lineage related to data being published as a result of data integration. The lineage will describe source and process/operation data
- lineage data a data level, meaning the property level of entities
- lineage data at the instance level of properties

This report will also include a specification of relating lineage as metadata to conceptual UML application schema at the property level.

§ 1.2 Target audience

Information modeller, API developer, IT architect.

§ 1.3 Introduction to document

Beschrijf hoe het document gelzezen moet worden, welke hoofdstukken voor wie bedoeld zijn, enz...

Analysis
Requierments and approach
Expert Sessions
Information model
Conclusions and recommendations

§ 1.4 Working proces

This document, lineage model, is developed in the WaU project ..... The principal working method is agile development through several High5 sessions on use cases of increasing complex information content and ....

2. Metadata MIM extension

Lineage information is considered as metadata in relation to elements in a productmodel. For this reason we consider lineage as metadata and will provide a common solution to relate metadata to semantics of a productmodel.

2.1 Requirements

The lineage model provides lineage information as metadata at the property level .

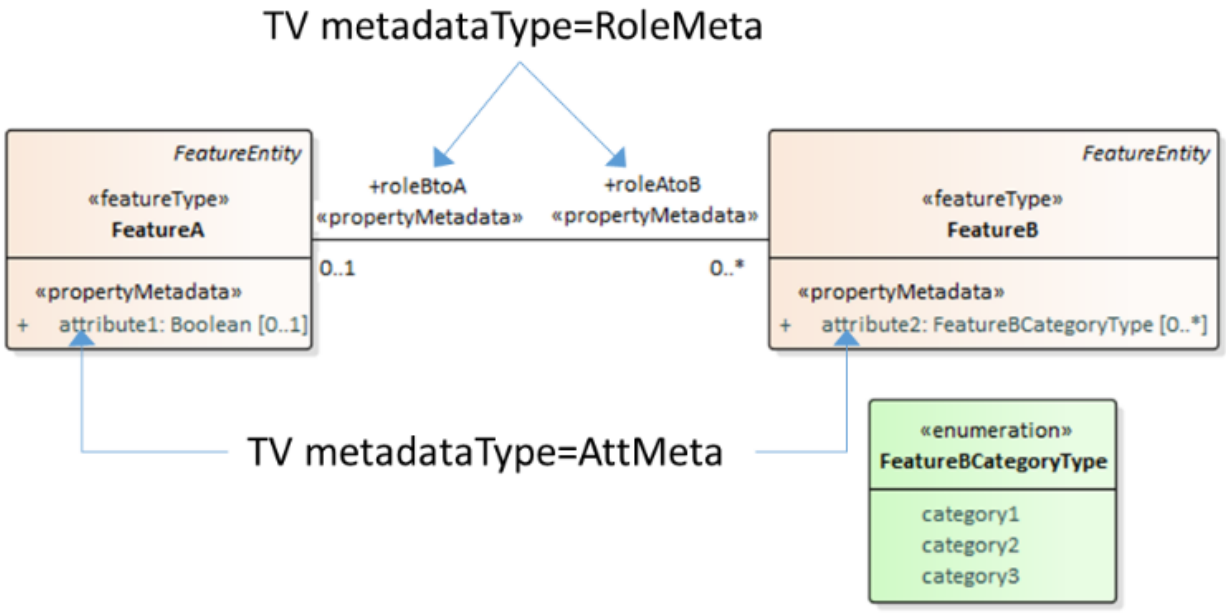
Requierments of the metadata in this respect are the following:

1. metadata at the property level of objecttypes
2. metadata at the instance level of properties
3. metadata can optionally be published
4. metadata model does not have effect on a productmodel

ad 1,2: UML-MIM does not have a construct to bind (meta)data at properties, i.e. attributes or association ends. It is therefore necessary to extend UML-MIM.

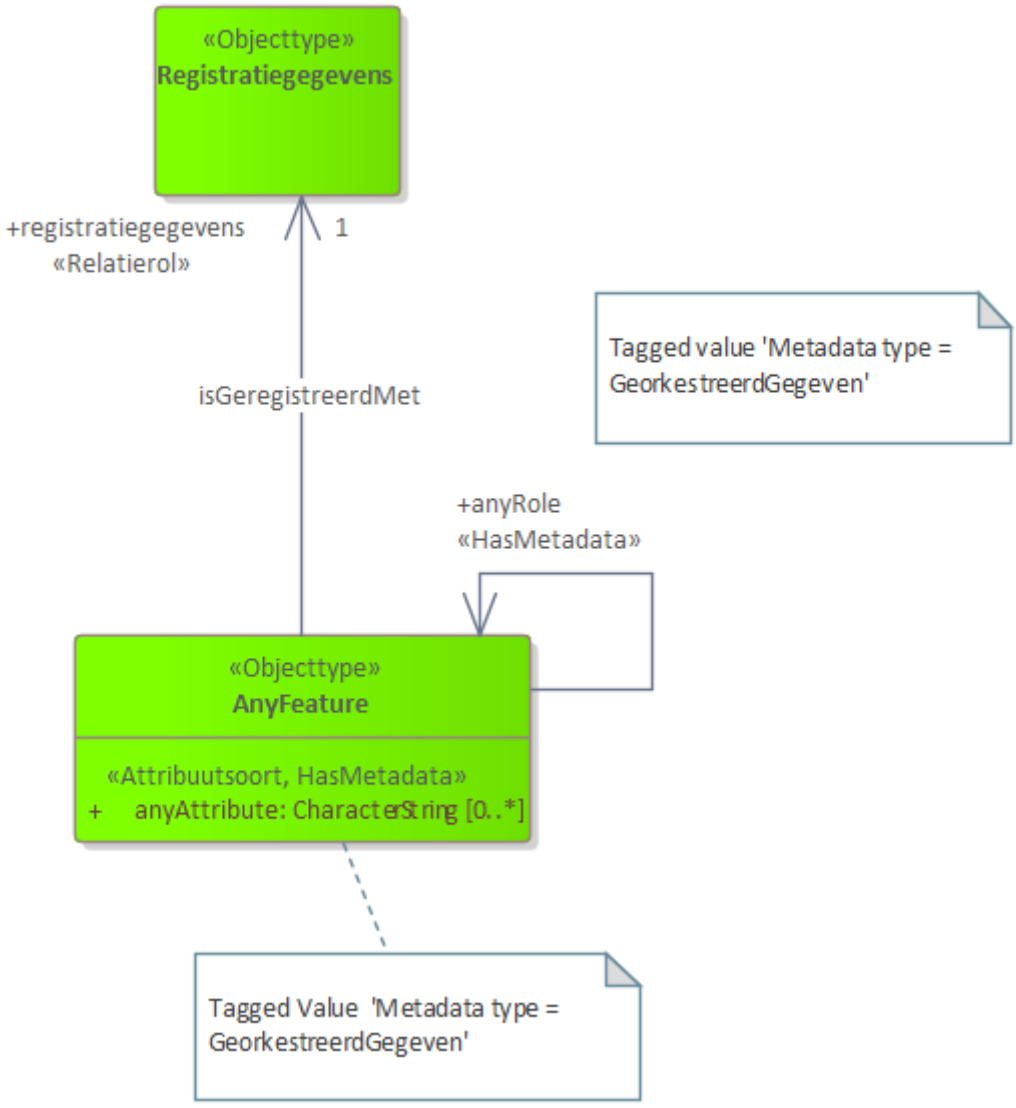
ad 3,4: The metadata or in this case lineage model is modelled independently from the product model. Both are 'loosely' connected.

An approach to these requiremnst can be found in the [Property-Stereotype-for-Metadata]. This publication presents an UML extension through a stereotype «`propertyMetadata`» and a tagged value `metadataType`. An example is presented below.



Figuur 1 Example showing use of stereotype: Properties with stereotype «`HasMetadata`» reference metadata types via tagged value `metadataType`. Source:[Property-Stereotype-for-Metadata]

The above presented pattern is slightly adapted by changing the sterotype name from propertyMetadata to a general «`HasMetadata`». The pattern is presented below on a general class AnyFeature representing any possible feature type in a productmodel.



Figuur 2 Metadata expressed at the conceptual level of a productmodel

Explanation and definition.

Stereotype «HasMetadata»

> **Definition «HasMetadata»**
> This element has associated metadata.

Description: The stereotype specifies that this information element has metadata associated. This is a specification at the conceptual level. A related tagged value `Metadata type` specifies the association to specific metadata.

Tagged Value: Metadata type

> **Definition Metadata type**
> Name of the target class of associated metadata.

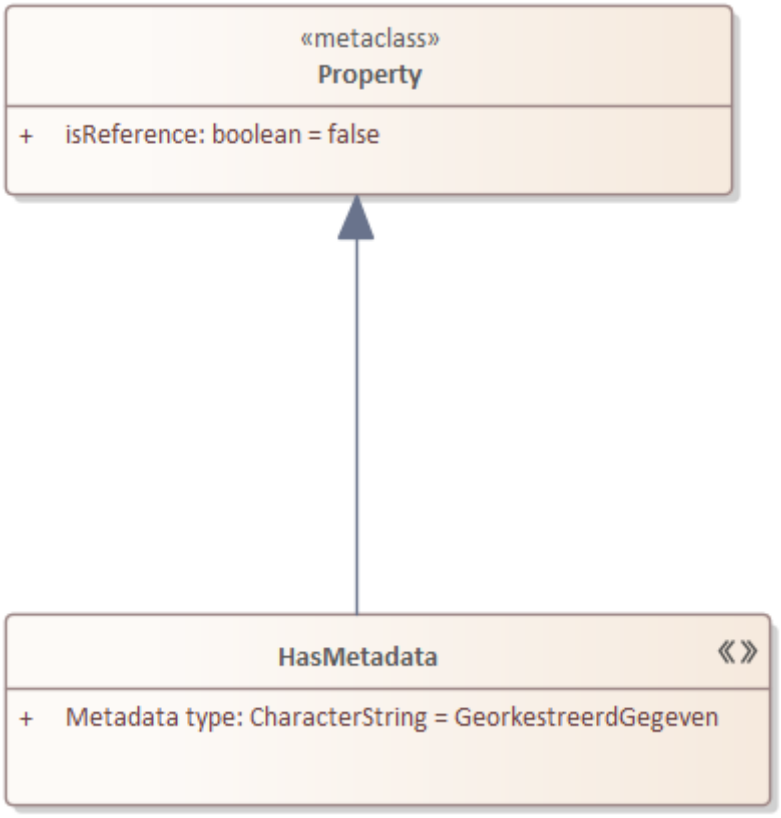
Description: This tagged value specifies the relation to a specific target class of or in a specific metadata model. This target class serves as the receptor defining the type of metadata that is associated.

In figure 2 the value for `Metadata type = GeorkestreerdGegeven` is only an example and may differ according to the required metadata class that relates to the specified metadata.

§ 2.2 MIM-UML extension

The defined stereotype and related tagged value lead to an extension of the MIM model. As the extension for now is independent from MIM the extension is described both as an extension of the UML metamodel and the MIM-UML model.

Extension on UML



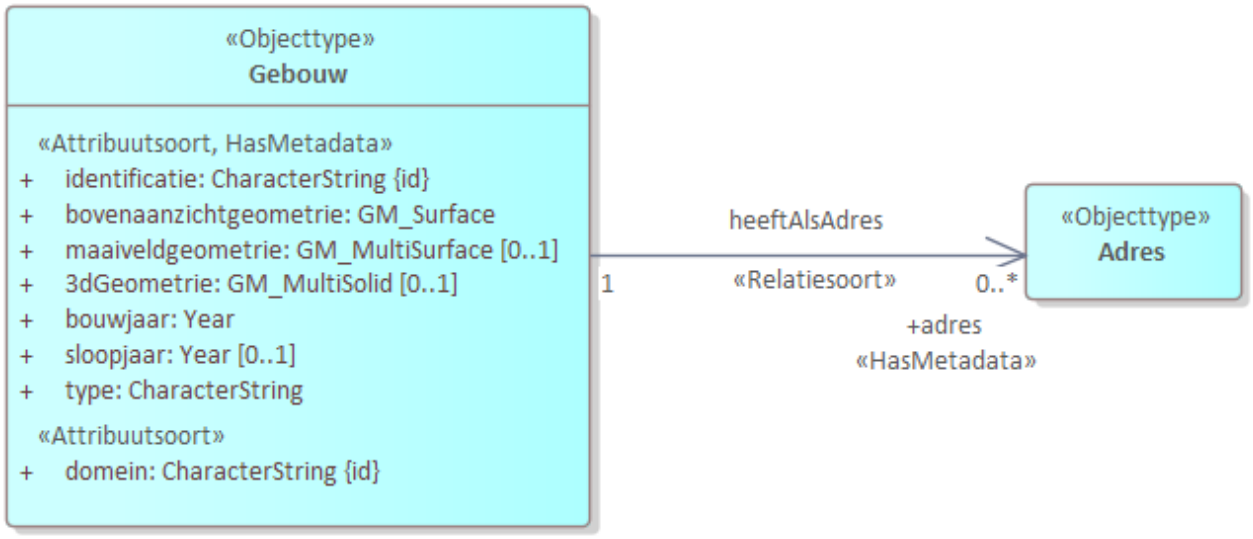
Figuur 3 Stereotype «`HasMetadata`» as extension on UML metaclass Property

This UML profile is implemented in a EA MDG Technology named HasMetadata_mdg.

Extension on MIM-UML

Figuur 4 Todo: Stereotype «HasMetadata» as extension on MIM-UML metaclasses

Example of this Metadata profile applied in a MIM-UML model.

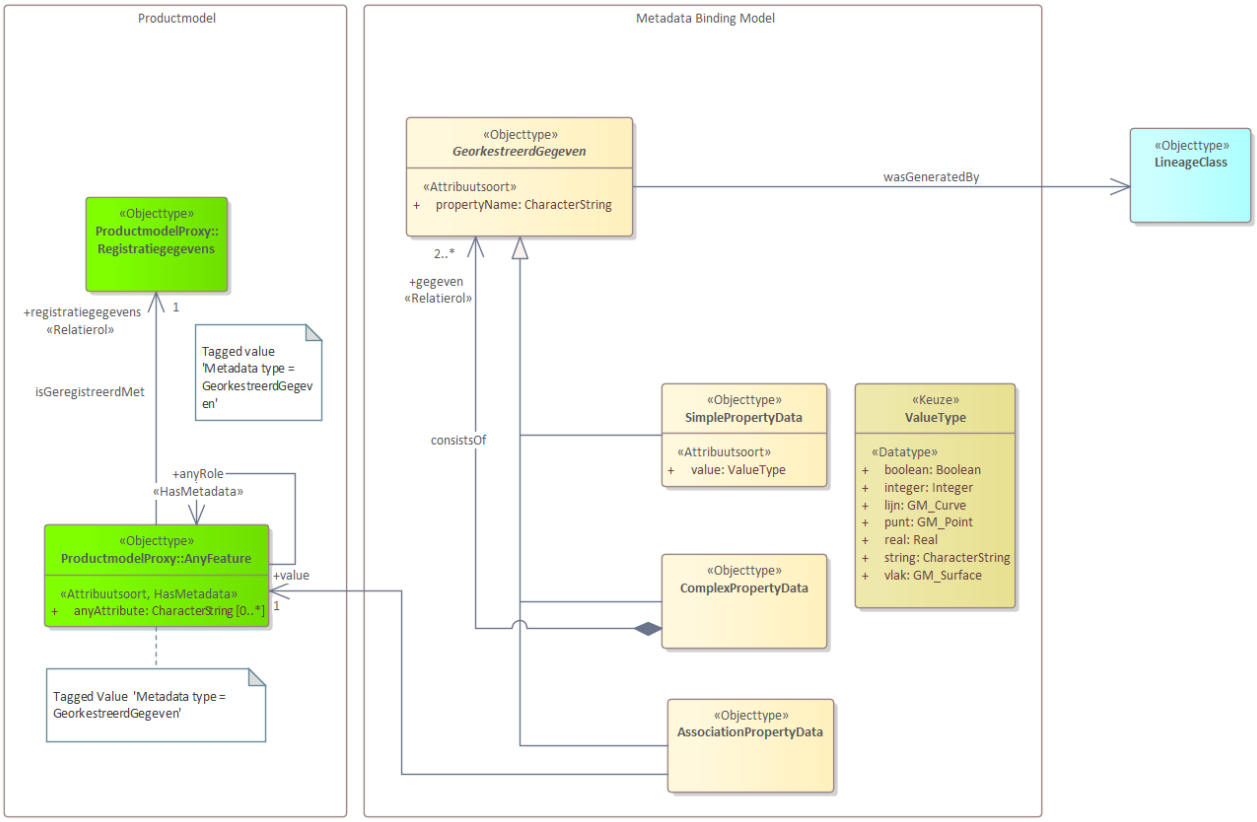


Figuur 5 Example of Stereotype «HasMetadata» applied in MIM-UML model

§ 2.3 Implementation, encoding.

The challenges regarding the implementation of this conceptual metadata construct vary across different encoding formats. For example in XML a construct exists to add attributes to XML elements, which provides a means to give attributes to properties, in this case reference to a metadata object. In JSON this does not exist. The «HasMetadata» stereotype is therefore extended with a model transformation that objectifies properties of a productmodel. The objectified properties are not part of the productmodel but serve as a loosly coupled intermediate binding mechanism between data and metadata.

The UML model of this pattern is presented below.



Figuur 6 Pattern for the Metadata binding model. Objectification of properties.

Explanation of diagram.
The Metadata binding model serves as a objectification of the properties of the productmodel. Each property (attributes and association roles) is transfomed to an instance of the objecttype `GeorkestreerdGegeven` having its name in `propertyName`. Depending on the type of property (or its associated value type) a SimplePropertData, ComplexPropertyData or the AssociationPropertyData is chosen. SimplePropertyData for properties with a unstructured valuetypes, ComplexPropertyData for structured valuetypes and AssociationPropertyData for association roles. The value of a property is transformed to the `value` attribute or the `value` association role.

In this model there is no explicit information link between a property and its value of the productmodel and its objectified `PropertyData` and value instance. To bind data of the productmodel to instances of 'PropertyData' is bij convention. The convention is that instances from objecttypes of the Productmodel and its properties and values by convention are bindend to instances of PropertyData and values in the Metadata Binding Model having the value for `propertName` and `value` equal to the name of the property and its value.
So without specifying specifically for the three subtypes of `GeorkestreerdGegeven` the convention rule is: `Productmodel.property = GeorkestreerdGegeven.propertyName AND Productmodel.property.value = GeorkestreerdGegeven.value`

§ 3. Data element

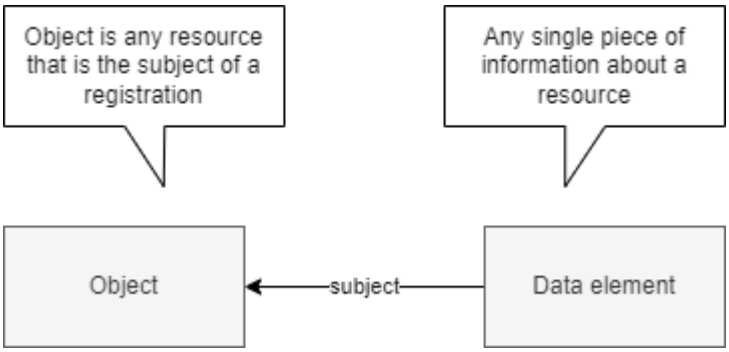
The first notion we introduce is that of a single statement or data element. Information objects we consume to draw insights from or base decisions on are comprised out of multiple atomic units of information, each captured by one attribute or association. Consider the following information object in JSON:

```
{
  "omschrijving": "Laan van Westenenk 701, 7334DP Apeldoorn",
  "plaatsnaam": "Apeldoorn",
  "identificatie": "0200200000075716",
  "isHoofdadres": true,
  "huisnummer": 701,
  "postcode": "7334DP",
  "straatnaam": "Laan van Westenenk"
}
```

Or, in RDF:

```
<http://example.com/id/adres/0200200000075716> a gebouw:Adres;
  gebouw:postcode "7334DP";
  gebouw:huisnummer 701;
  gebouw:identificatie "0200200000075716";
  gebouw:omschrijving "Laan van Westenenk 701, 7334DP Apeldoorn";
  nen3610:registratiegegevens <http://example.com/base/registratiegegeven
  gebouw:straatnaam "Laan van Westenenk";
  gebouw:plaatsnaam "Apeldoorn";
  gebouw:isHoofdadres true .
```

Each data element can have its own lineage; its own algorithm to determine the given value. By treating these data elements as object; allows us to describe and provide information about them. For instance to state that `{ "plaatsnaam": "Apeldoorn" }` is derived from a data element in the BAG. In the following diagram both the object and the data elements are modelled as an objecttype, as they are things; objects, we are interested in describing.



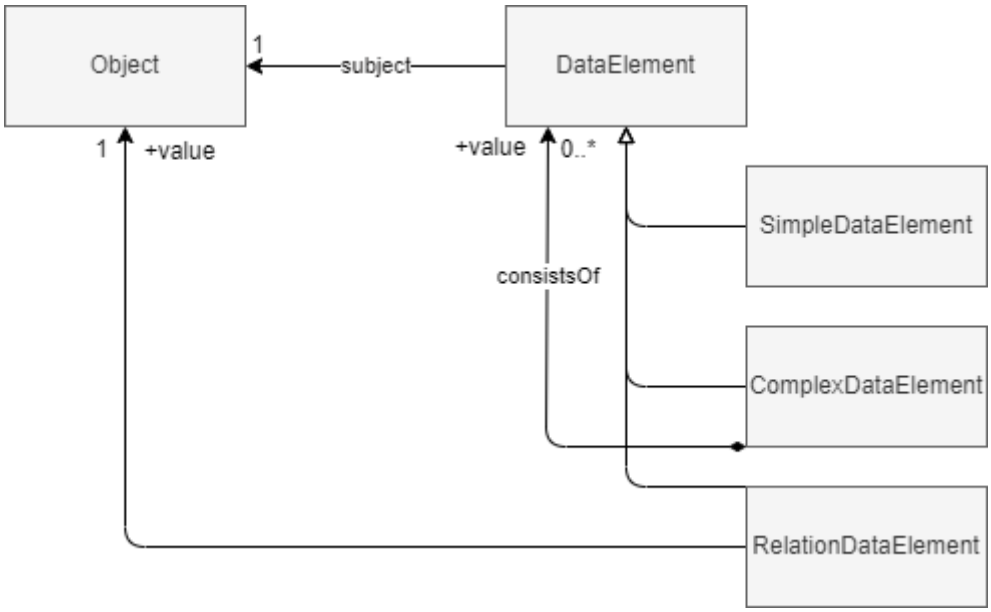
Figuur 7

A data element is identified by it's 1) subject 2) value and 3) property e.g.

```
{
    "property": "plaatsnaam",
    "value": {
        "stringValue": "Apeldoorn"
    },
    "subject": {
        "identificatie": "0200200000075716",
        "domein": null
    }
}
```

Term	Definition
object	An object is any resource that is the subject of a registration.
data element	A data element is any singular piece of information about a resource.
subject	A subject is an object a data element is about.
property	A property is the characteristic of the subject a data element is about.
value	The value of a data element.

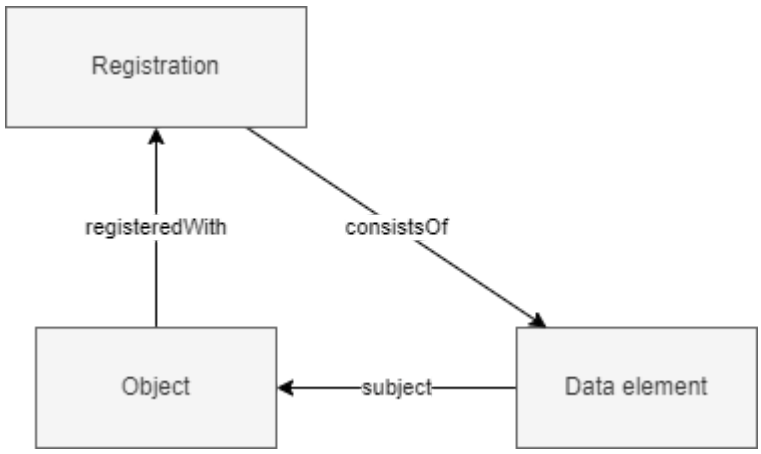
Several subtypes of data elements are identified; but these are omitted in the following examples for simplicity's sake.



Figuur 8

A data element can be described from different point of views. The perpective the lineage model takes is one of provenance. Questions to be answered include, for instance, where, how and by whom, is the statement `Building G0200.42b3d39246840268e0530a0a28492340 has construction date 2006` created? Other models could address other aspects of a data element; for instance whether it is subject to an examination of correctness (which could be relevant for the BAG: https://catalogus.kadaster.nl/bag/nl/page/InOnderzoek) or what is the quality of a data element.

A data element is part of the information object describing an object; but this is omitted in the following examples.



Figuur 9

This gives a basic model applicable for many usecases. This lineage model add to this a way to connect one data element to one or more other data elements it is derived from. The latter are called source data elements, the former orchestrated data elements. For the source data elements it

is important to know which registry or dataset it is retrieved from so that users can interpret the source data element in the context it was published in. This also allows allows users to find related information.

4. Conformiteit

Naast onderdelen die als niet normatief gemarkeerd zijn, zijn ook alle diagrammen, voorbeelden, en noten in dit document niet normatief. Verder is alles in dit document normatief.

5. Lijst met figuren

A. Index

A.1 Begrippen gedefinieerd door deze specificatie

§ A.2 Begrippen gedefinieerd door verwijzing

§ B. Referenties

§ B.1 Normatieve referenties

[Property-Stereotype-for-Metadata]
Reference not found.

↥