Data Sharing Agreements

Table of Contents

1. Introduction

In order for organisations to exchange data, it is important that they have a common language in which they can communicate. The data exchange also requires some formal agreement between a requesting party (client) and a delivering party (supplier) to determine what information is shared and how this information should be structured. This document introduces a standard for Data Sharing Agreements which provides a way to meet these needs.

Two important requirements for this standard are interoperability and simplicity. Interoperability is required in order to apply the standard in any environment, regardless of the information systems being used by both parties. Simplicity is required to ensure that it can be understood and used without the need for extensive IT knowledge. To this end, only things that are essential should be added to the vocabulary for the Data Sharing Agreements.

This standard makes use of existing Semantic Web standards by the World Wide Web Consortium (W3C), such as RDF and RDFS. It provides a general, coherent vocabulary for Data Sharing Agreements. This can be used by a client as a framework to create a definition of how data should be delivered by the supplier. This enables the supplier to perform a data delivery according to their Data Sharing Agreement.

The next chapter of this document describes the general principles for sharing data. Each subsequent chapter will introduce a new model for data sharing for a specific purpose. These models can be used either individually or combined with each other, depending on the use case. The appendix contains a list of all concepts used in this document, as well as a triple representation of all examples in this document.

Created and published by Semmtech BV, 2018, version 1.0.. www.semmtech.com

2. Sharing

First of all, the general principles on which these Data Sharing Agreements are based should be covered: the use of open standards, unique naming of resources and tracking of different versions.

2.1 Open data standards

To facilitate the exchange of data, regardless of the platforms used by client and supplier, it is important that an open, non-proprietary file format is used. In this way, both parties can read and interpret the data that is being exchanged. The Data Sharing Agreement (DSA) standard will make use of RDF. In this format, all information is stored in the form of triples: object – relation – object. All objects and relations are defined by a URI (unique resource identifier) or a literal (some fixed value).

2.2 Naming of resources

URIs are an identification mechanism which allows for a unique identification of resources. New URIs are often created by applications by combining a namespace and a local Unique Identifier (UID) of a record. The namespace (e.g. 'http://example.org/definition/') tends to indicate the location of the dataset and can be abbreviated (e.g. 'def:'). The UID (e.g. '904A80') can be based on existing identifiers or codes, such as the identifier of a table row, or generated automatically for each new piece of data. The full URI is then created by adding the UID to the namespace, such as 'http://example.org/definition/904A80' or, using an abbreviated namespace, 'def:904A80'.

Names of concepts are defined in the Data Sharing Agreements by the rdfs:label relation. Descriptions are recorded with rdfs:comment.[1] These two properties are part of the RDF Schema vocabulary that is recommended by the W3C[2]. Instead of creating new properties for this purpose, these existing ones are reused in the Data Sharing Agreements.

2.3 Version control

For versioning, each published dataset can receive a version number. Using the example namespace from the previous paragraph, this would be 'http://example.org/definition/version_01/' for example. Individual elements in the dataset do not receive version numbers.

2.4 Namespace of Data Sharing Agreements

This document offers a few modules for various use cases. Each of these modules offers a complete solution and can therefore be used individually. It is also possible to use several modules alongside each other for use cases that require it. The namespace of the vocabulary used by the modules is: http://w3id.org/dsa/.

[1] The full URIs of these two relations are: *http://www.w3.org/2000/01/rdf-schema#label* and *http://www.w3.org/2000/01/rdf-schema#comment*

[2] The W3C (World Wide Web Consortium) is an organisation that facilitates the development of standards for the Web.

3. Data Delivery Agreement

Use case: "As a client I want my suppliers to deliver data for asset management, required for my own information systems: data about designed and built assets, their coherent structure (parts) and their properties."

The Data Delivery Agreement vocabulary can be divided into two parts. One part provides the client with a way to model how the data should be delivered. This part is called the data definition. The second part provides the supplier with a way to deliver the data according to this definition. These are discussed in separate subsections.

3.1 Data definition

Every conceptual physical object can be a whole for (or part of) another conceptual physical object (e.g. 'car' and 'wheel'). In this way, partonomies of objects can be created. At the same time, the 'is specialization of' relation can be used between objects (e.g. 'car' and 'SUV') to create a taxonomy of objects. Partonomies and taxonomies are, in this use case, required for navigation.

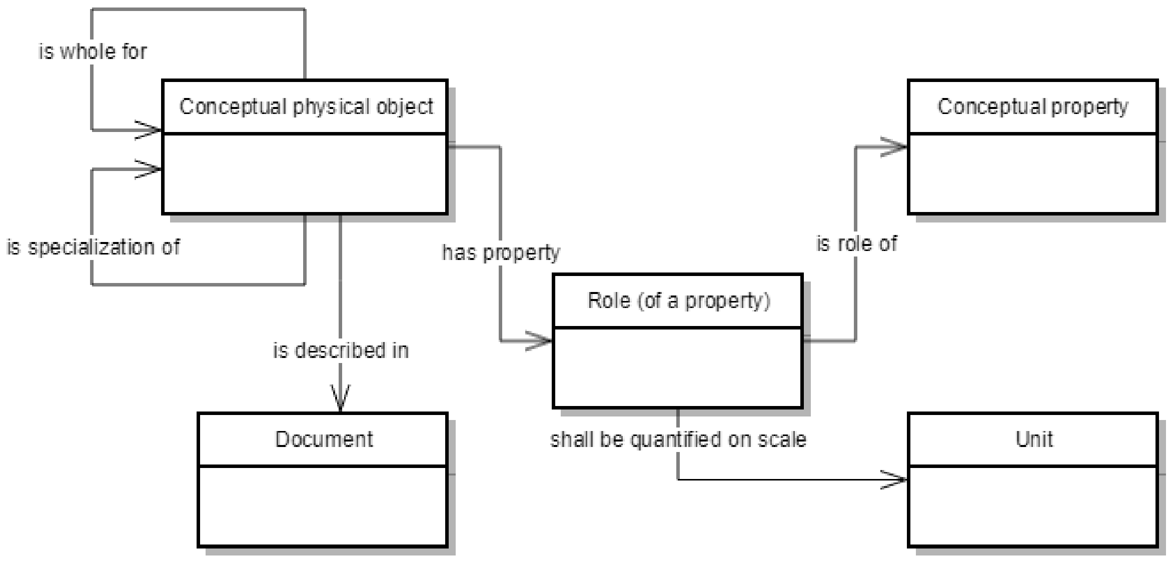

Figure 1: Data definition schema

In order to state that a certain property for an object needs to be filled out, both a conceptual property and a role of a property (i.e. the property *of* something) are needed. This can be modelled in the following way: a car has property 'height of a car', which is a role of the conceptual property 'height'. In this way, it is possible to say things specifically about the height of a car (for example that it shall be quantified on scale of 'centimetre'). The 'height of a car' can thus be said to require registration in 'metre', without restricting how the height of another object should be measured.[3] Figure 2 shows this example according to the schema presented in Figure 1.

[3] See also 'ISO 16354 Guidelines for knowledge libraries and object libraries' and 'ISO15926:2003 Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 2: Data model'.

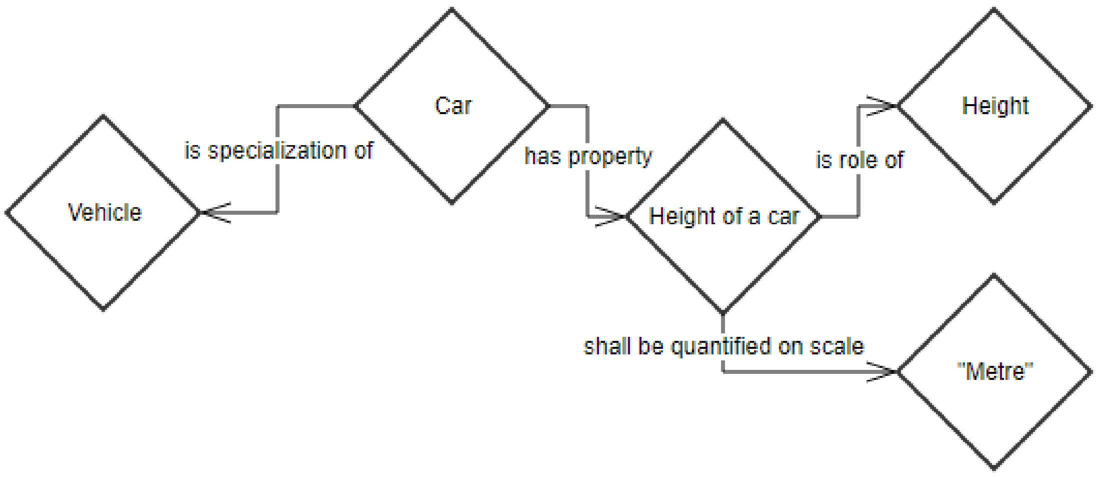

Figure 2: Data definition instances

Not all properties are measured with a unit (e.g. colour). Therefore, a role can alternatively be linked to a collection of possible values. Figure 3 is an extension of the schema of Figure 1: ‘collection’ and ‘value’ have been added here. To illustrate, possible values for the role ‘colour of a car’ could be modelled in the following manner: ‘colour of a car’ has as options ‘car colours’ (a collection). This collection would then contain several colours (various instances of ‘value’ in the schema). Finally, these colour values could be linked to the conceptual property ‘colour’ by stating that they are qualifications thereof. This example is shown in Figure 4.

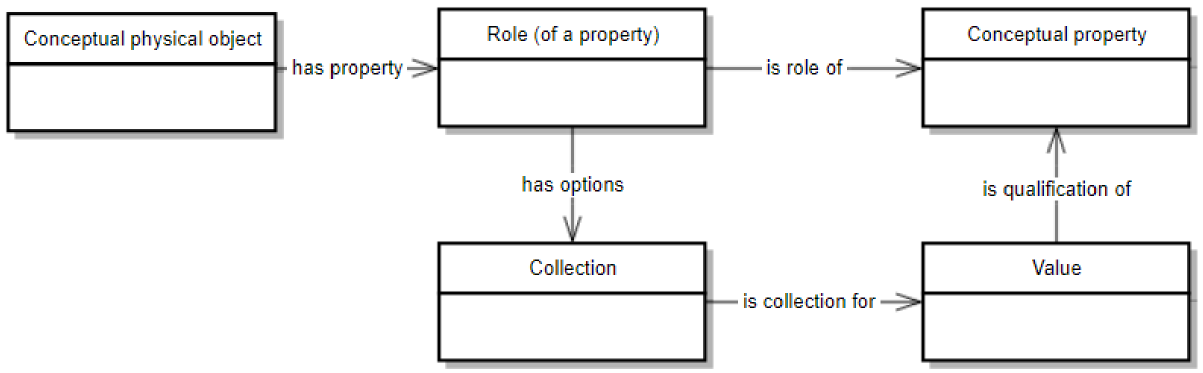

Figure 3: Using collections in the data definition

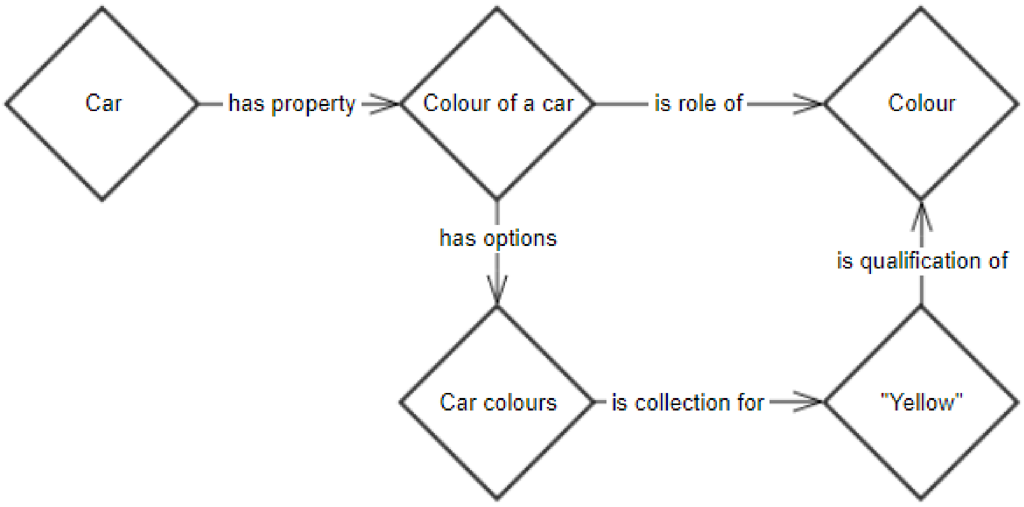

Figure 4: Instances for a collection

The third and final option is to define the datatype of the required values. This applies to data that does not use a specific unit of measurement or a collection of possible values. This is the case for texts or amounts without a unit such as 'two pieces'. In these cases, it suffices to connect the required datatype for a value to the role. The XML schema datatypes (xsd) ontology can provide this functionality[4]. For example: 'manufacture date of a car' has datatype 'xsd:date' (see Figure 5).

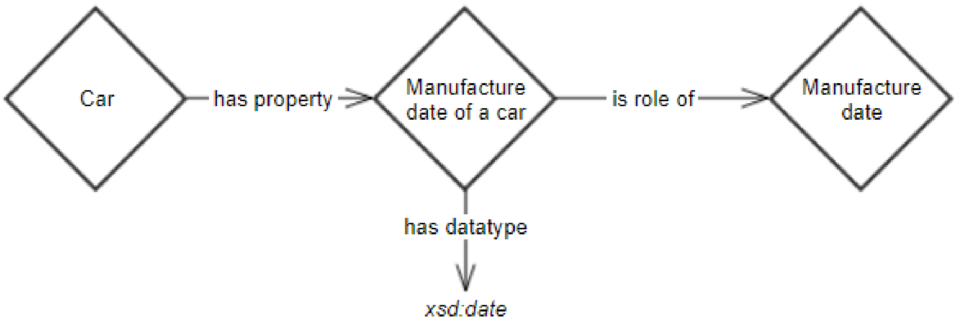

Figure 5: Instances for a role with a datatype

The DDA vocabulary also allows for documents to be linked to a conceptual physical object. This is done through the relation 'is described in'. Next to the document's name, its file location can be stored by linking this to the document, e.g. 'document 1' has file location "Document1.docx".

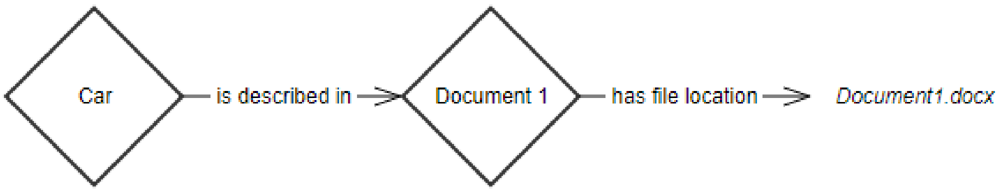

Figure 6: Instances for an object with a document

[4] A list of possible datatypes can be found here: https://www.w3.org/TR/xmlschema-2/#built-in-datatypes .
The namespace of the XSD ontology is: http://www.w3.org/2001/XMLSchema# .

3.2 Data delivery

When a supplier returns information about specific objects, this can be expressed using the schema presented in Figure 7. Individual objects can be classified as either functional (e.g. a technical drawing of a car) or materialized (an actual car). These two types behave similarly in the DDA vocabulary and are therefore shown as a single entity in Figure 7. An individual object can have an individual property such as 'height of my car', which is a 'height' (conceptual property). Individual objects should be linked to a conceptual object using the 'is a' relation.

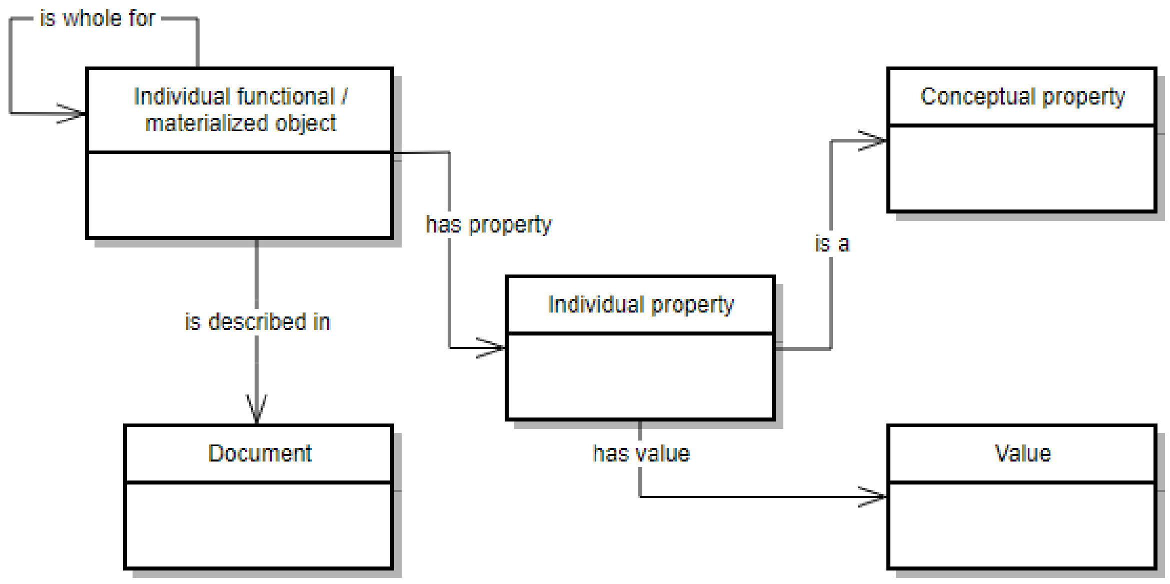

Figure 7: Data delivery schema

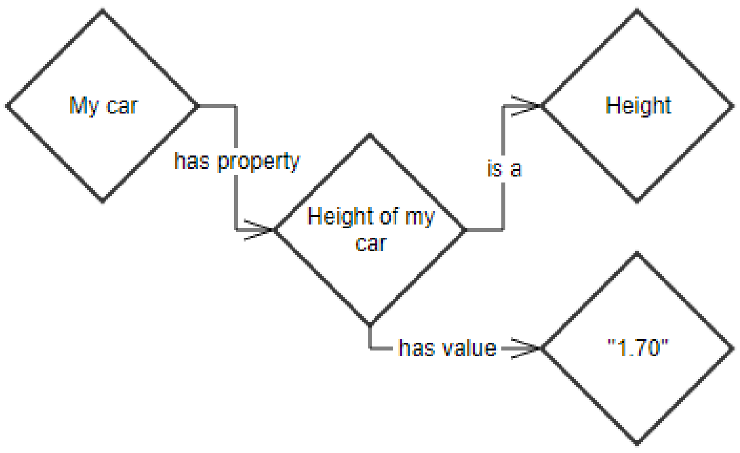

Figure 8: Data delivery instances

Values that are associated with a unit, such as 'height of my car', are recorded with the xsd:double datatype (e.g. "1.70"^^xsd:double). Values from a collection are referred to using a URI (e.g. def:Yellow). If instead a datatype has been specified in the data definition, the recorded value in the data delivery must adhere to that datatype.

Cardinality is not covered in the vocabulary of the Data Delivery Agreement. Adding semantics for cardinality would add more complexity to the vocabulary. This would go against the primary goal of this standard to keep it as simple as possible. On top of that, cardinality constraints cannot fully ensure completeness either. Parties entering into such an agreement should therefore agree that all fields within the set definition are mandatory. Upon receipt of the data, its completeness can be validated. Fields that have not been filled can then be checked to investigate why they have not been filled.

It is possible to extend this basic vocabulary with more concepts if a certain use case requires it. The next sections will provide a few standard building blocks for other use cases, but anyone is free to add any project-specific definitions.

4. Mapping application records for DSAs (e.g. for CAD, GIS, P&ID)

Use case: ”As part of a supply chain I want to receive data about one or more designs and their source documents/files.”

For individual functional objects, such as drawings of an object, it can be required to share information on their representations. Possible examples include a CAD drawing or spatial information in a Geographic Information System (GIS). Such an agreement can be modelled in a similar way to documents in the Data Delivery Agreement.

First, the client defines conceptual physical objects (including any whole/part and specialization relations). Afterwards, the supplier can make a data delivery according to the schema in Figure 9.

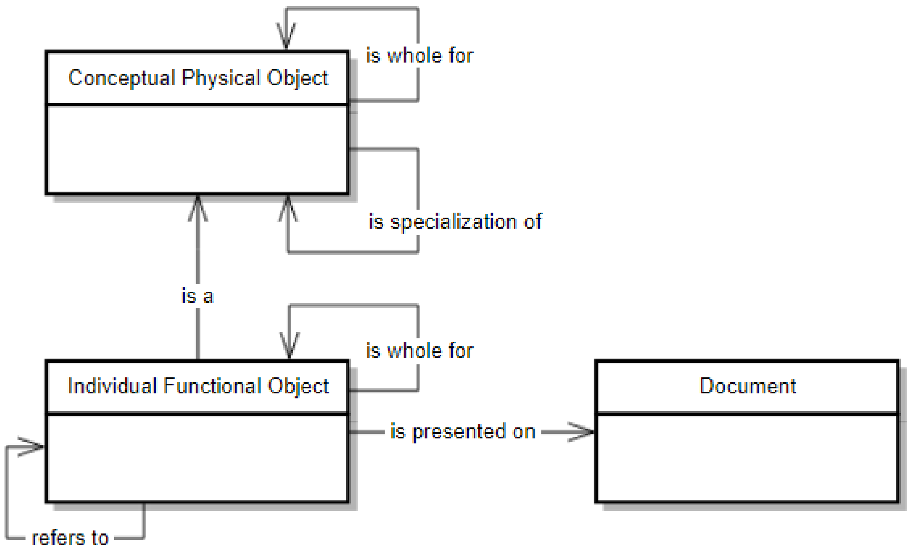

Figure 9: Mapping application records

An individual functional object can be presented on a document. For example, individual functional object ‘CAD drawing of this car’ is presented on a ‘CAD document’. This individual functional object in turn refers to another individual functional object that is independent of any representational form (e.g. ‘drawn car’). Each different representation of this ‘drawn car’ has its own pair of an individual functional object and document that refer to the ‘drawn car’. Thus, the characteristics of each representation can still be distinguished from each other if desired.

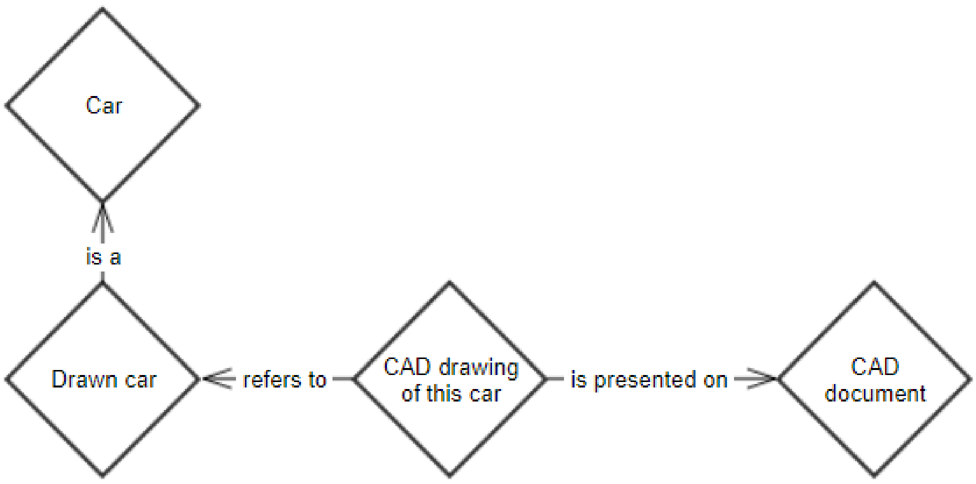

Figure 10: Application record instances

5. Distributing contract requirements

Use case: "As a client I want to distribute requirements for assets (either types of assets or project specific individual assets) to my suppliers in order for them to populate their requirements management systems efficiently and without manual failures."

The basic DDA vocabulary from Chapter 3 does not contain any provision for specifications. Two new classes are introduced for this purpose: 'conceptual specification' and 'individual specification'. As part of the data definition, various objects (conceptual physical objects and/or individual functional objects) and individual specifications should be defined by the client (e.g. 'Car' and 'Mandatory presence of airbags'). Each individual specification is linked to an object (either conceptual or individual), as can be seen in Figure 11.

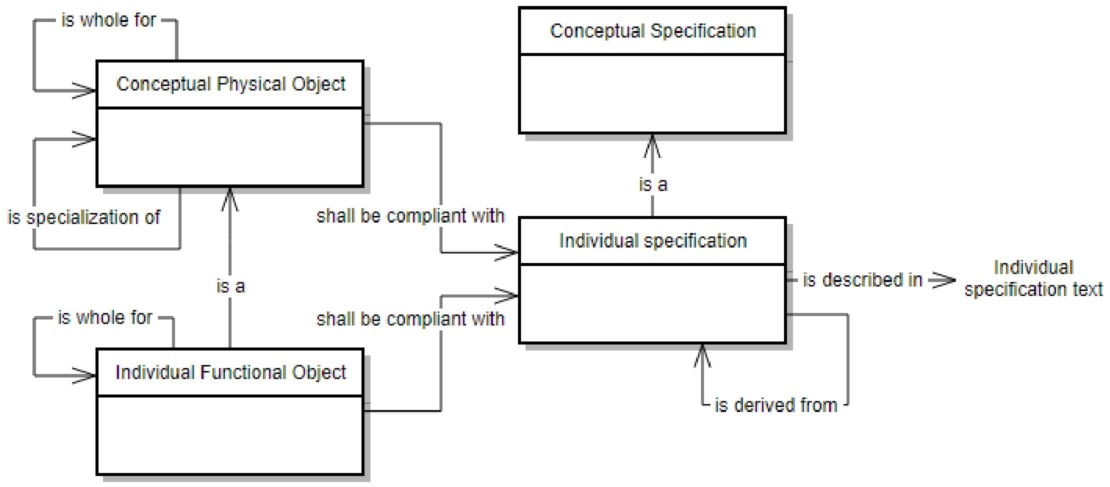

Figure 11: Adding specifications to objects

An individual specification has an individual specification text that contains the actual text of the specification, stored as a string. Moreover, an individual specification may be classified according to a conceptual specification, e.g. 'Safety specifications'. Indicating that an individual specification has been derived from another specification is also possible. This situation is shown in Figure 12, where the specification 'Mandatory presence of airbags' is derived from 'Safety of a car'.

Only actual individual specification texts should be sent by the delivering party, as no status or version information is sent with it. Using this agreement, the receiving application will know that it should only use the specification texts that were included in the most recent data delivery.

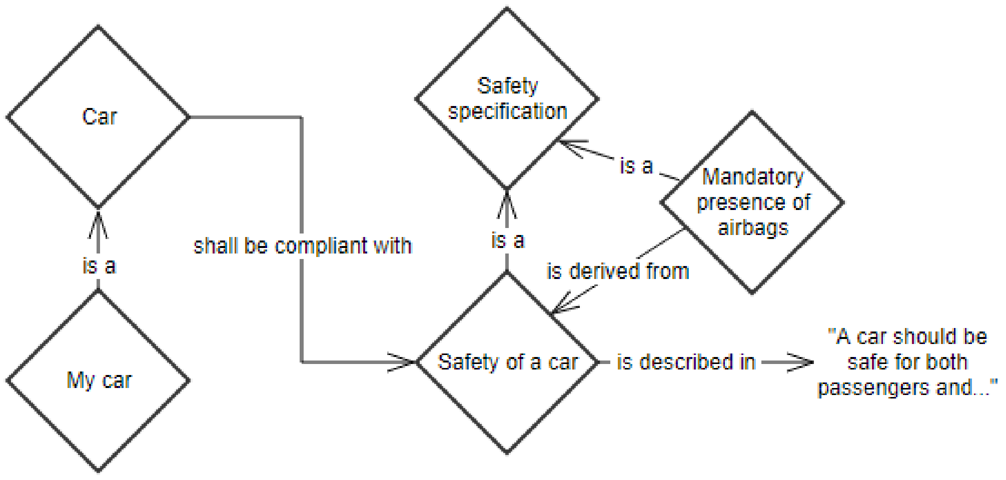

Figure 12: Instances for specifications

Appendix 1: Classes for the Data Sharing Agreements

Table 1: All classes for the data sharing models

Class	**Description**
Module 1: Data Delivery Agreement	
dsa:Collection	A collection of values (e.g. the possible colours of a car)
dsa:ConceptualPhysicalObject	A conceptual physical object (e.g. a car)
dsa:ConceptualProperty	A property (e.g. height)
dsa:Document	A document that contains information on an object
dsa:IndividualFunctionalObject	A representation of a certain object (e.g. drawing of my car)
dsa:IndividualMaterialObject	An actual physical object (e.g. my car)
dsa:IndividualProperty	A property of a specific object (e.g. height of my car)
dsa:Role	A role of a property (e.g. height of a car)
dsa:Unit	A unit of measurement
dsa:Value	A value of a dsa:Collection or an actual value for an individual property
Module 2: Mapping application records	
dsa:ConceptualPhysicalObject	… see description module 1
dsa:IndividualFunctionalObject	… see description module 1
dsa:Document	… see description module 1
Module 3: Distributing contract requirements	
dsa:ConceptualPhysicalObject	… see description module 1
dsa:ConceptualSpecification	A conceptual specification (e.g. safety specifications)
dsa:IndividualFunctionalObject	… see description module 1
dsa:IndividualSpecification	A specification for an object

Appendix 2: Properties for the Data Sharing Agreements

Table 2: All properties for the data sharing models

Property	Description
Module 1: Data Delivery Agreement	
dsa:isA	Classifies individual concepts to a conceptual concept
dsa:hasDatatype	Defines the datatype of a role
dsa:hasOptions	Links a collection to a role
dsa:hasValue	Links an individual property to a value
dsa:hasFileLocation	Links a document to its file location
dsa:hasProperty	Link from an object to a role or individual property
dsa:isCollectionFor	Link from a collection to (one of) the values within the collection
dsa:isDescribedIn	Link from an object to a document
dsa:isQualificationOf	Link from a value of a collection to a conceptual property
dsa:isSpecializationOf	Specialization relation between objects, to create a taxonomy
dsa:isRoleOf	Links a role to a conceptual property
dsa:isWholeFor	Whole/Part relation between objects, to create a partonomy
dsa:shallBeQuantifiedOnScale	Defines the required unit of measurement for a role
Module 2: Mapping application records	
dsa:isA	… see description module 1
dsa:isPresentedOn	Link between an individual functional object and the document that presents it
dsa:isSpecializationOf	… see description module 1
dsa:isWholeFor	… see description module 1
dsa:refersTo	Link between one representation of an individual functional object
Module 3: Distributing contract requirements	
dsa:isA	… see description module 1
dsa:isDescribedIn	Link between a specification and its text
dsa:isDerivedFrom	Link between specifications to indicate that one is derived from another
dsa:shallBeCompliantWith	Link between an object and an associated specification
dsa:isSpecializationOf	… see description module 1
dsa:isWholeFor	… see description module 1

Appendix 3: Turtle/RDF notation for the examples

Prefixes used in the examples

```
@prefix dsa:  <http://w3id.org/dsa/> .
@prefix def:  <http://example.org/definition/> .
@prefix del:  <http://example.org/delivery/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
```

Data Delivery Agreement

```
#--------------- DDA Definition ---------------

def:Vehicle          a      dsa:ConceptualPhysicalObject ;
       rdfs:label    "Vehicle" .

def:Car              a      dsa:ConceptualPhysicalObject ;
       rdfs:label                  "Car" ;
       dsa:isSpecializationOf      def:Vehicle ;
       dsa:hasProperty             def:HeightOfACar ;
       dsa:hasProperty             def:ColourOfACar ;
       dsa:hasProperty             def:ManufactureDateOfACar ;
       dsa:isDescribedIn           def:Document1 .

def:HeightOfACar     a      dsa:Role ;
       rdfs:label    "Height of a car" ;
       dsa:isRoleOf                def:Height ;
       dsa:shallBeQuantifiedOnScale def:Metre .

def:Height           a      dsa:ConceptualProperty ;
       rdfs:label    "Height" .

def:Metre            a      dsa:Unit ;
       rdfs:label    "Metre" .

def:ColourOfACar     a      dsa:Role ;
       rdfs:label    "Colour of a car" ;
       dsa:hasOptions def:CarColours ;
       dsa:isRoleOf  def:Colour .

def:CarColours       a      dsa:Collection ;
       rdfs:label            "Car colours" ;
       dsa:isCollectionFor   def:Yellow .

def:Yellow           a      dsa:Value ;
       rdfs:label           "Yellow" ;
       dsa:isQualificationOf def:Colour .

def:Colour           a      dsa:ConceptualProperty ;
       rdfs:label    "Colour" .

def:ManufactureDateOfACar  a   dsa:Role ;
       rdfs:label           "Manufacture date of a car" ;
       dsa:isRoleOf         def:ManufactureDate ;
       dsa:hasDatatype      xsd:date .

def:ManufactureDate  a      dsa:ConceptualProperty ;
       rdfs:label    "Manufacture date" .

def:Document1        a      dsa:Document ;
       rdfs:label           "Document 1" ;
       dsa:hasFileLocation  "Document1.docx" .
```

```
#--------------- DDA Delivery ---------------

del:MyCar             a      dsa:IndividualFunctionalObject ;
       rdfs:label            "My car" ;
       dsa:isA               def:Car ;
       dsa:hasProperty       del:HeightOfMyCar ;
       dsa:hasProperty       del:ColourOfMyCar ;
       dsa:hasProperty       del:ManufactureDateOfMyCar .

del:HeightOfMyCar     a      dsa:IndividualProperty ;
       rdfs:label     "Height of my car" ;
       dsa:isA        def:Height ;
       dsa:hasValue   "1.70"^^xsd:double .

del:ColourOfMyCar     a      dsa:IndividualProperty ;
       rdfs:label     "Colour of my car" ;
       dsa:isA        def:Colour ;
       dsa:hasValue   def:Yellow .

del:ManufactureDateOfMyCar a dsa:IndividualProperty ;
       rdfs:label     "Manufacture date of my car" ;
       dsa:isA        def:ManufactureDate ;
       dsa:hasValue   "2018-01-01"^^xsd:date .
```

Mapping application records for DSAs

```
def:Car               a      dsa:ConceptualPhysicalObject ;
       rdfs:label     "Car" .

del:DrawnCar          a      dsa:IndividualFunctionalObject ;
       rdfs:label     "Drawn car" ;
       dsa:isA        def:Car .

del:CADDrawingOfThisCar   a  dsa:IndividualFunctionalObject ;
       rdfs:label            "CAD drawing of this car" ;
       dsa:refersTo          del:DrawnCar ;
       dsa:isPresentedOn     del:CADDocument .

del:CADDocument       a      dsa:Document ;
       rdfs:label     "CAD document" .
```

Distributing contract requirements

```
def:Car               a      dsa:ConceptualPhysicalObject ;
       rdfs:label                   "Car" ;
       dsa:shallBeCompliantWith     def:SafetyOfACar ;
       dsa:shallBeCompliantWith     def:MandatoryPresenceOfAirbags .

def:MyCar             a      dsa:IndividualFunctionalObject ;
       rdfs:label     "My car" ;
       dsa:isA        def:Car .

def:SafetyOfACar      a      dsa:IndividualSpecification ;
       rdfs:label            "Safety of a car" ;
       dsa:isA               def:SafetySpecification ;
       dsa:isDescribedIn     "A car should be safe for both passengers and ..." .

def:MandatoryPresenceOfAirbags a  dsa:IndividualSpecification ;
       rdfs:label            "Mandatory presence of airbags" ;
       dsa:isA               def:SafetySpecification ;
       dsa:isDerivedFrom     def:SafetyOfACar ;
       dsa:isDescribedIn     "A car should have an airbag ..." .

def:SafetySpecification   a  dsa:ConceptualSpecification ;
       rdfs:label     "Safety specification" .
```