# Standard-oriented ontology export of domain catalogues from data dictionaries

Sebastian Schilling[1], Christian Clemen[1]

[1]Dresden University of Applied Sciences, Faculty of Spatial Information, Friedrich-List-Platz 1, Dresden, 01069, Germany

**Abstract**

The use of standardised data dictionaries is becoming increasingly important in the construction industry. Property servers such as the buildingSMART Data Dictionary (bSDD), the freeBIM property server or the BIM Portal Germany have therefore been developed for the uniform provision of data dictionaries. These are used in the Building Information Modelling (BIM) process to describe objects with predefined properties. At the same time, data dictionaries are being linked across domain boundaries. This may be achieved with the help of Semantic Web technologies. The data dictionaries must also be made available as ontologies, as it has already been done with the ifcOWL ontology for the Industry Foundation Classes (IFC), for example. The aim must be for property servers to be able to export the uniform, and in future also cross-domain, data dictionaries as ontologies in the Web Ontology Language (OWL). Initial approaches for the bSDD have already been developed. Our paper presents an approach for an OWL export based on the ISO 12006-3 meta standard. The metamodel described therein is used to organise information about construction works. It is implemented in our self-developed open source property server editor datacat. The export is explained with the 2016 version of the ISO 12006-3 standard, while the implementation of the new 2022 version is currently in progress. The two versions of the standard will be compared and their changes presented.

**Keywords**

data dictionary, property server, ontology export, ISO 12006-3, BIM, GIS

## 1. Introduction

While the Geographic Information System (GIS) industry has established, common standards for data dictionaries for many years, the development of common dictionaries in the construction industry has only just begun. To date, each discipline involved in construction has its own, inconsistently structured data dictionaries for describing construction objects. As a result, even though they mean the same thing, different terms are often used in planning or to describe objects that actually exist on the construction site. Another problem is that the data dictionaries are usually not centralised and freely accessible, which can also lead to inconsistencies due to different versions within a data dictionary. With the spread of Building Information Modelling (BIM), uniformly structured data dictionaries are becoming increasingly important. The data dictionaries should be made freely available on the web and uniformly searchable so that BIM software can use them. This has led to the development of property servers, the first of such servers are already in use. Examples include the buildingSMART Data Dictionary (bSDD), the Austrian freeBIM property server and the property module of the German BIM Portal.

At the same time, the linking of data dictionaries is becoming increasingly important in the context of Linked Data and the Semantic Web, as it makes it easier to find and display relationships. The creation of ontologies from data dictionaries and their export must therefore also be a goal in the development of property servers. At present, approaches for exporting dictionary content in the Resource Description Framework (RDF) as a semantic graph are only available for bSDD. We want to explore how dictionaries can be exported as ontologies and present our standard-oriented approach.

The construction industry has established the ISO 12006-3 [1] standard, which provides a metamodel for the organisation of information about construction objects. This standard is also used as the basis for property servers, such as the bSDD. Our research aims to define an ontology export based on this

*LDAC2024 - 12th Linked Data in Architecture and Construction Workshop, June 10–14, 2024, Bochum, Germany*
sebastian.schilling@htw-dresden.de (S. Schilling); christian.clemen@htw-dresden.de (C. Clemen)
0000-0003-3718-4598 (S. Schilling); 000-0002-5807-7698 (C. Clemen)

standard, so that the ontologies of different domains may later be linked more easily. In addition, the standardised Dublin Core[1] metadata schema will be used to describe the exported ontology concepts. We restrict the ontology so that we stay within the domain of the concepts described and enable data exchange in BIM. This also helps with the automated creation of ontologies from the property server. The modelling decisions in each domain have already been made by the experts when creating the data dictionary in the property server, so that classes, properties and values are available in a standardised schema. The advantage of this is that users of the property server do not need any knowledge of formal ontologies. We have implemented this approach in our open source property server called datacat. As a new version of the standard ISO 12006-3:2016 with a changed metamodel has been published in 2022, we also show the differences between the versions and give an outlook on how to update the property server and migrate the existing data dictionaries.

In the following Section 2 we first explain existing approaches (Section 2.1) before introducing our datacat property server (Section 2.2). In the Section 3 we explain our approach to ontology export and its implementation in datacat. The changes in the new metamodel of the ISO 12006-3 standard are described in Section 4. Finally, the results and limitations are discussed in Section 5, before a summary and outlook are given in Section 6.

## 2. Related Work

The use of standardised data dictionaries as ontologies is not a new concept and is already used in the health sector, for example, to link health data[2]. However, ontologies are not yet widely used in BIM and GIS. In BIM, this is also due to the fact that data dictionaries have not yet been used in a standardised, cross-domain and centralised way. In the construction industry, there is no technical standard schema against which all data dictionaries are defined. In the GIS domain, however, there are the widely used standards of the ISO 19000 series, in particular ISO 19101-1 [2] and ISO 19110 [3], which define the structure of feature catalogues. The feature catalogues are used to describe the characteristics of geographic data sets. They therefore have a very similar function to data dictionaries in BIM. The property servers are a tool intended to solve the problem of inconsistent data dictionary descriptions in the construction industry. According to [4], a property server can be defined as follows: "Running in the background for the user, a property server provides harmonised, unique and machine-readable properties and offers a standard for the parameterised description of digital construction components." The major innovation of property servers in the construction industry is that data dictionaries are made available and updated centrally. This makes them much easier to use in a consistent way than decentralised distribution via files that quickly become outdated and circulate in different versions.

### 2.1. Existing Ontology Export Approaches

Exporting entire data dictionaries, individual classes and/or properties or property sets from property servers as resources or as ontologies in RDF has already been investigated with several approaches.

A dynamic OWL OpenAPI, an interface for querying the properties of IFC elements, which returns the properties and property sets in the Web Ontology Language (OWL) is implemented in [5]. This is done by querying the properties of the IFC element via the bSDD API, mapping the query result to a small ontology and finally outputting it via an OpenAPI. Property sets are defined as OWL object properties whose domain is the IFC class. The individual properties of the property set are defined as OWL datatype properties. In principle, this approach is not only applicable to the IFC classification, but can also be generalised to other classifications of bSDD with adaptations. This approach focuses on standardising the output as OpenAPI. The ontology used for mapping is not specified and does not appear to be standardised. In addition, the approach has only been tested on IFC classifications.

[1] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[2] https://bioportal.bioontology.org/ontologies/SNOMEDCT

Another approach [6] mentions that bSDD concepts have been transformed into OWL resources, firstly to standardise the concepts and secondly to be able to use the multilingual description of the concepts. How the transformation is done is not described. However, this paper shows how the data dictionaries exported from property servers can be used as an ontology. The concepts exported from the bSDD are linked to the IFC ontology and other ontologies for exchange requirements, and are used to create and check rules to semantically validate IFC models.

There is no standard way to generate RDF graphs using the bSDD API is described in [7]. They give an example of what an RDF graph of a concept might look like. They also describe a 'conceptType' property, which can take the concept types from the ISO 12006-3:2016 [1] standard as values. We will use this typing in a similar way in our approach. [7] also note that the properties from bSDD can be used in the ontology as object and datatype properties, which is how we will define them.

An approach for creating product descriptions using the ISO 12006-3:2006 meta standard is presented in [8]. They propose a hierarchical metamodel with four layers. Layer M0 is the kernel layer and contains the ISO 12006-3:2006 meta concepts. Building on this, domain-specific concepts are created as instances of the meta concepts in Concept Library Layer M1. The Product Kernel Layer M2 defines additional concepts that help to convert abstract concepts from layer M1 into instances. In the Product Instantiation Layer M3, products are instantiated using layers M1 and M2. The advantage of this approach is that the abstract concepts can be kept strictly separate from the product instances. Layers M0 and M1 are similar to the structure of property servers, where the ISO 12006-3 metamodel is used to define concepts in data dictionaries. We follow a similar approach to export the data dictionaries as ontologies from the property server.

The export from the property server can also be done first in another format, e.g., JSON, and then translated into the Web Ontology Language (OWL) using a converter, as described in [9].

Alternatively, there is no export and a link is only created from the ontology to the property server via the GUIDs of the concepts. In their approach, [10] do not export features from the bSDD. Instead, they add a feature to the classes of their Building Product Ontology (BPO) in which the corresponding globally unique identifier from the bSDD can be entered to create a link to the non-RDF data. However, this makes the semantic interpretation of the bSDD information with tools difficult.

The bSDD has a preview feature for its API[3] to return API requests as RDF in RDF/XML and TURTLE syntax. However, this only applies to the '/api/Class/v1' endpoint. It returns the requested class with its metadata in RDF. All associated properties and subclasses can also be returned via query attributes.

The presented approaches differ, among other things, in what is to be exported. On the one hand, there are approaches for exporting an entire data dictionary or a sub-catalogue as an ontology, and on the other hand, there are approaches for exporting individual feature sets for specific classes. Both variants make sense depending on the use case and should therefore be implemented in our software. Table 1 summarises the presented approaches with their similarities and differences.

**Table 1**
Overview of existing research in property server export

| Paper | Export Level | Source | Method | Exported Concepts |
|---|---|---|---|---|
| [5] | OWL | bSDD | OpenAPI | IFC property sets + properties |
| [6] | OWL | bSDD | unknown | concepts |
| [7] | OWL | bSDD | unknown | unknown |
| [10] | no export | bSDD | reference property with bSDD GUID | nothing |
| [? ] | RDF | bSDD | OpenAPI | classes + properties |

A description of how features from the construction industry can be represented in the Semantic Web is given in [11]. Among other things, they describe how simple and complex features can be created in graphs and defined using different approaches. The approach we want to use is to define properties as instances of the OWL concepts *owl:ObjectProperty* and *owl:DatatypeProperty*. Unlike other approaches,

[3]https://app.swaggerhub.com/apis/buildingSMART/Dictionaries/v1#/

such as using *rdf:Property* or *owl:AnnotationProperty*, this approach also allows the modelling of OWL class restrictions, which we need for our research.

## 2.2. datacat Editor for Managing and Publishing Data Dictionaries

The well-known property servers such as the bSDD, the freeBIM property server and the BIM Portal Germany are hosted by an organisation through which other organisations can make their data dictionaries freely available with read-only access. The creation and editing of data dictionaries tends to take a backseat. For this reason, as described in [12], we have developed a property server with an editor to serve these work processes. Our property server can also be hosted by anyone and used locally on a computer. Our datacat[4] is an open source software consisting of the property server in the backend and a web frontend for editing and creating data dictionary entries. Like bSDD, the property server is based on the ISO 12006-3:2016 [1] standard. This standard provides a metamodel for describing concepts in the construction industry. In addition, the ISO 23387 [13] standard is used for data templates. Figure 1 describes the datacat layers and how they are linked. The concepts are stored in a Neo4j graph database and organised using the meta standard. On top of this is the business logic that processes the information according to the user's input. The backend provides its functionalities via a GraphQL interface and accesses the graph database internally.

The datacat editor is a browser-based interface that communicates with the backend via the GraphQL interface. The interface has been tailored to the needs of users who have already used data dictionary creation during development. Thanks to the open source approach, the editor can be adapted for other disciplines or a completely different editor can be developed for the backend. In order to have read access to the concepts, the implemented interface requires users to register once in the datacat editor. An administrator of the data dictionary must grant the user write access to add, edit and delete concepts. The user can view the concepts separately by type or as a simplified tree structure and follow the links between the concepts. Concepts are added in two steps. First, the concepts are created, then the relationships between the concepts are created. The data dictionaries can currently be exported in three ways:

1. The concepts and their relationships are exported as a CSV file.
2. Individual concepts are queried as JSON via the GraphQL interface.
3. The data dictionary or parts of it are exported as an ontology in TURTLE syntax.

The third approach will be presented in the following section.

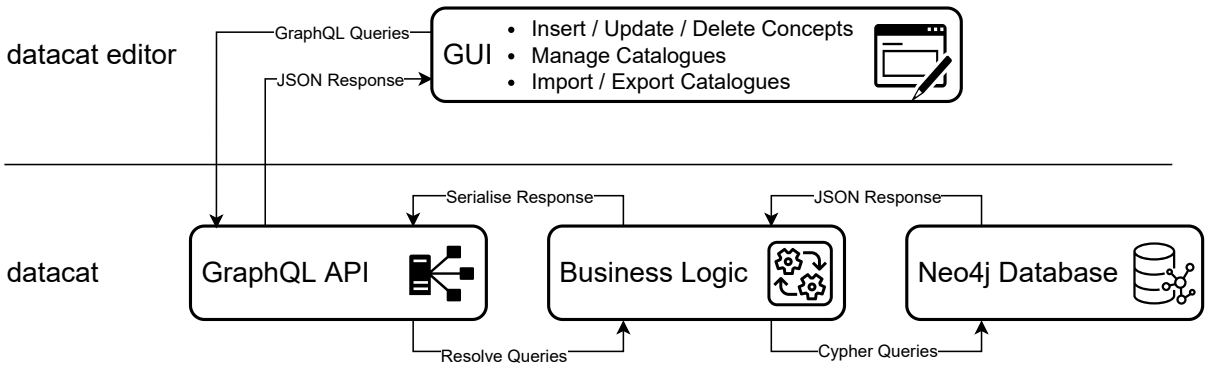


**Figure 1:** The datacat layers and their interconnections

[4]https://github.com/dd-bim/datacat

## 3. Standard-oriented Ontology Export of Domain Catalogues

The concepts created by the expert groups in datacat should be usable in BIM projects across disciplines, i.e. in different domains and software systems, for the classification and attribution of objects. We need the concepts as an ontology so that we can link the domains in a Semantic Web context. The structured storage of the concepts in the property server is already graph-based and based on the ISO 12006-3:2016 meta standard, so the step to an ontology is not far away. In our approach, we want to implement the meta standard as a meta schema in order to be able to export all concepts in a standardised, standard-oriented way as an ontology. This will have the advantage that we can link ontologies from different domains easily because the concepts have the same structure. Figure 2 shows an example of the different levels from the metamodel to the classified object instance. The figure is based on the M0 to M3 layer structure from [8]. The meta concept layer contains the ISO 12006-3:2016 metamodel. The metamodel is instantiated in concept dictionary layer when domain experts provide their expertise as concepts in the property server. In the ontology layer, the concepts are translated into OWL to make the information available in the Semantic Web technology stack. In the instance layer, the resulting ontology can be used to describe instances in RDF.

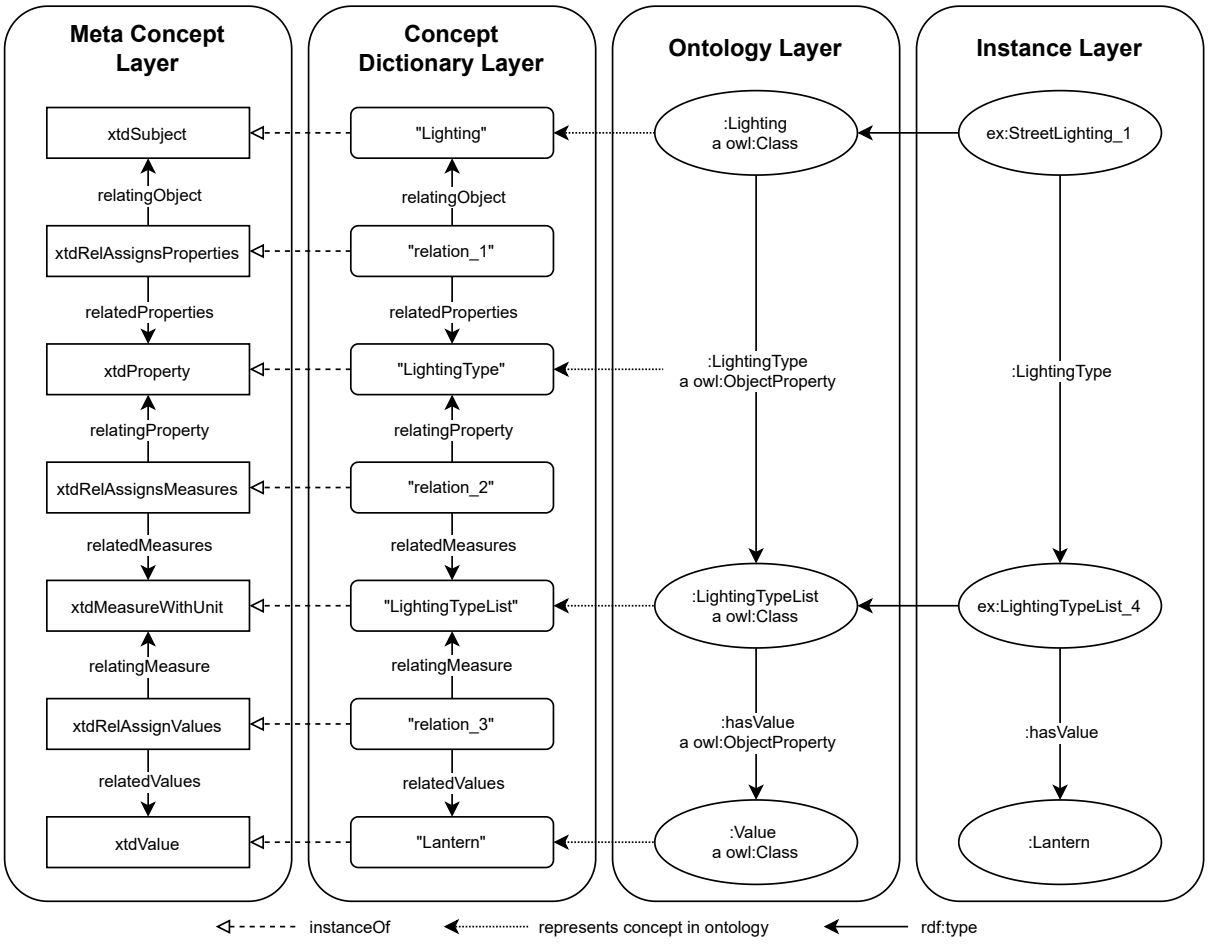


**Figure 2:** Process layers from the meta concept to the object instance (based on [8])

The GraphQL interface of the property server is used for the ontology export to query the concepts. The interface already has predefined queries for the datacat editor for all concept types and relationships, which can also be used by our exporter.

As a result of our research and implementation, the full data dictionary can be exported from the property server as an ontology, or a concept of type *xtdSubject* with its associated features, predefined value lists and units can be exported by specifying the name.

Figure 3 shows the four main steps that are performed in our programme. First, a query is formulated to the GraphQL interface. Its response in JSON format is interpreted and then transformed into OWL concepts according to the concept type of the metamodel. Finally, the Turtle syntax is used to output the ontology. The process is described in detail in the following.

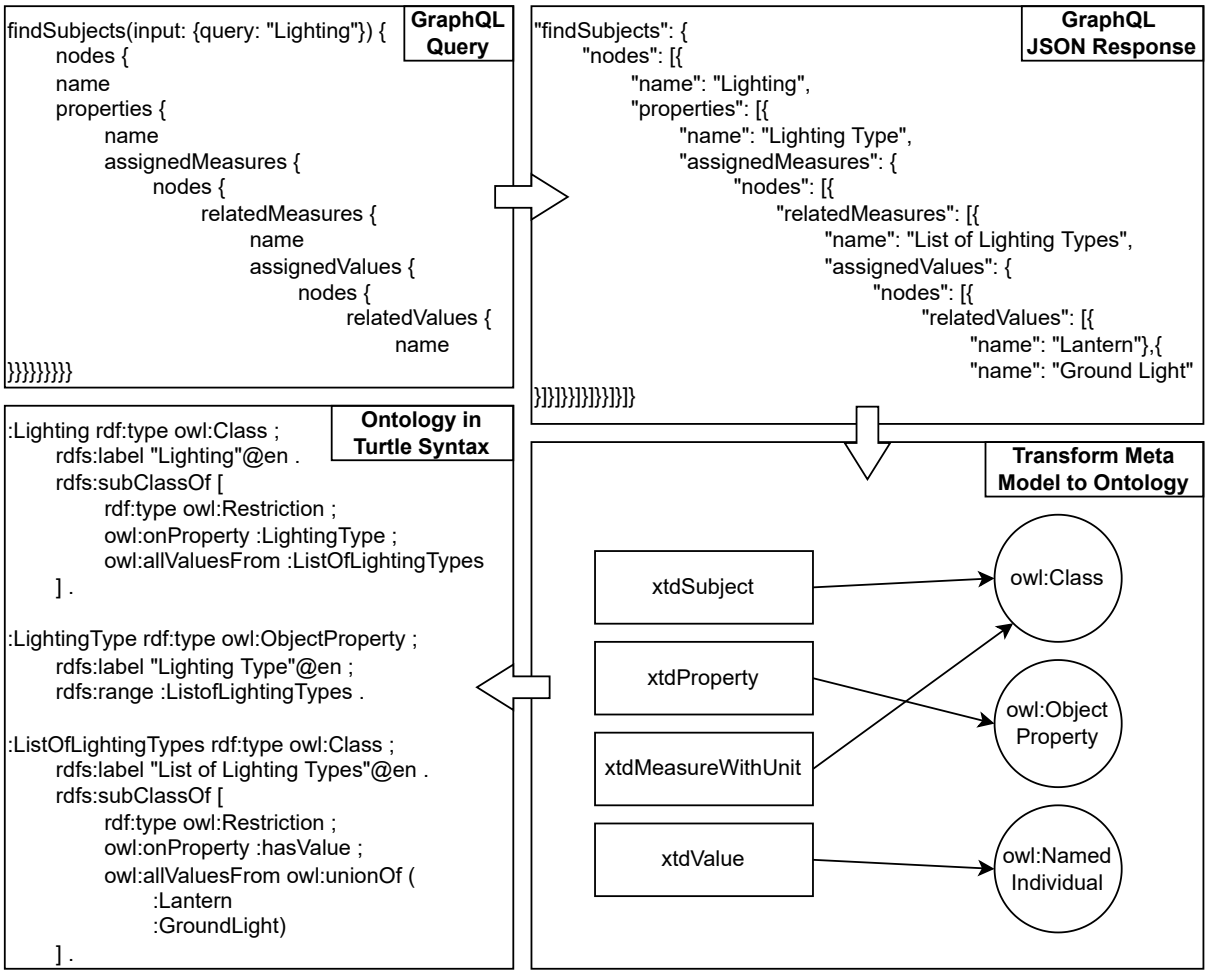


**Figure 3:** Data processing sequence for ontology export from the property server

If a specific concept shall be exported, first a GraphQL query *findSubjects* is sent to the property server with the name as an input parameter. As GraphQL queries can be nested at will, the various concepts linked by relationships can be output in the same query, along with the simple attributes. The returned data in JSON format is mainly mapped to standardised ontology elements and added to the result graph as a triple. If *xtdProperty* concepts are found, they are linked to the *xtdSubject* concept as *owl:ObjectProperty*. Finally, the result graph is output to a file in Turtle syntax. We want to explain our design choices here.

The concepts are stored in the property server as instances of the classes of the ISO 12006-3:2016 metamodel. All concepts may also contain metadata. This includes, for example, the creation date, the creator, a globally unique identifier (GUID) and a description of the concept. We use the Dublin Core standard to store and use the metadata in the ontology. We have chosen this standard because it is also a globally recognised ontology and therefore well suited for comparing and linking ontologies. In order to maintain the reference to ISO 12006-3:2016, the attribute *dcterms:type* is used to append the concept type of the standard to each exported concept. This also contributes to a better comparability of the ontologies. Table 2 gives an overview of the metadata attributes used. Other metadata, such as *VersionID* and *VersionDate*, must be defined as attributes themselves, as no standardised ontology has been found for them.

By attaching the GUID from the property server as metadata to the ontology elements, the dictionary entry can be uniquely referenced.

**Table 2**
Standard-based metadata properties from other ontologies

| Metadata from datacat | Used ontology property |
|---|---|
| name | rdfs:label |
| creator | dcterms:creator |
| created | dcterms:created |
| modified | dcterms:modified |
| description | dcterms:description |
| id | dcterms:identifier |
| ISO 12006-3 concept type | dcterms:type |
| Prefixes | rdfs: http://www.w3.org/2000/01/rdf-schema#<br>dcterms: http://purl.org/dc/terms/ |

The name of the concept can be stored in the property server in multiple languages by appending the appropriate language tag to the string as it is entered. We use the name as an identifier to make the ontology easier to read. Using the GUID for URIs (Uniform Resource Identifiers) would be better because of its uniqueness, but it has been found that the identifiers of concepts imported into datacat are not always suitable for URIs because they contain forbidden characters. In addition, the uniqueness of names is already taken into account in datacat. We currently use the German name because not all concepts in the property server have an English name. However, when exporting, you can choose to use either the German or the English name. There are no conventions for naming concepts in datacat. To ensure that URIs are valid and easy to read, the names must be adapted to the standard naming conventions during the ontology export process. This means removing spaces and special characters, stringing together words with camel case and replacing German umlauts.

To ensure that the true name is retained in the ontology and that the names of other languages can be used, they are attached to the concept as *rdfs:label* with a language tag. This means that the concept can later be searched in different languages in the ontology and can be used internationally.

The exact implementation of each concept type of the metamodel in OWL is not definite due to the many possibilities of specification. Design decisions must therefore be made here. We have decided to start with *xtdSubject* as *owl:Class* and gradually integrate the related concepts. The *xtdProperty* concepts are modelled as *owl:ObjectProperty*. The approach of modelling properties in OWL as *owl:ObjectProperty* has been adopted from [11]. The properties are associated with *xtdMeasureWithUnit* concepts, which are modelled as *owl:Class*. The *xtdValue* and *xtdUnit* concepts are linked to these as instances of an *owl:Class :Value* respectively *:Unit* with the self-defined *owl:ObjectProperty hasValue* and *hasUnit*. The concrete values and units can each be a list from where a concept can be selected. We have initially defined *owl:ObjectProperty* ourselves here, as we still need to standardise it on the datacat property server side. Until now, domain experts have defined their own *xtdUnit* concepts for units each time they create a dictionary, which leads to redundant work and leaves room for errors. As units are usually standardised, there are collections of units such as the Ontology of Units of Measure[5] (OM) that can be used to provide predefined units. When implemented in datacat, the *hasValue* and *hasUnit* properties from the OM ontology can be used to further standardise ontology export.

Restrictions need to be applied so that only the relationships that are actually allowed in the data dictionary are mapped in the ontology. In OWL, this is achieved by using axioms. The *owl:ObjectProperty* gets the attributes *rdfs:domain* and *rdfs:range*, which restricts their use to the resprective classes. The *owl:Class* concepts receive a restriction via *rdfs:subClassOf*, which regulates the linking of one or more other concepts via a *owl:ObjectProperty*. Multiple linkable concepts are defined as *owl:unionOf* via the attribute *owl:allValuesFrom* as a range for the property. The example in Listing 1 shows the class axiom

[5]https://bioportal.bioontology.org/ontologies/OM

of a class *:Facilities*. Instances of this class can only be linked to instances of the classes *:Lighting*, *:Pole* and *:Sign* using the object property *:collects*.

```
:Facilities rdf:type owl:Class ;
        dcterms:type "xtdBag" ;
        rdfs:label "Facilities"@en;
        rdfs:subClassOf [
                rdf:type owl:Restriction ;
                owl:onProperty :collects ;
                owl:allValuesFrom owl:unionOf (:Lighting :Pole :Sign)
        ] .
```

**Listing 1:** Class axiom for the restriction of the relationship between two classes in Turtle syntax

In the context of describing classes with properties and values, as is done in datacat, the class axioms can be used to specify that only certain values can be assigned to an instance of a class via a property.

When creating data dictionaries in the property server, the concepts can also be grouped, which is defined in the ISO 12006-3:2016 metamodel with the abstract concept *xtdCollection* and its subclasses *xtdBag* and *xtdNest*. The collections can be nested arbitrarily. For example, in our datacat user interface, concepts from *xtdSubject* can be grouped under a concept *xtdBag*. These in turn can be grouped under another *xtdBag* to form a domain model. *xtdProperty* concepts can be grouped into property groups under *xtdNest*. In addition, all concepts can be assigned a *xtdExternalDocument* as an external reference. With these definitions we can map a hierarchy often used in data dictionaries. Restrictions also help here to map the structures in the ontology.

If the entire data dictionary is to be exported as an ontology, the collection concepts are defined as *owl:Class* and have a restricted o*wl:ObjectProperty :collects* for the grouping, as shown in Listing 1. The definition of *xtdExternalDocument* concepts is done in the same way, only with *:documents* as *owl:ObjectProperty* for linking the described concepts.

## 4. Update datacat to the new version of ISO 12006-3:2022

A revised version of ISO 12006-3 [14] was published in 2022. [8] have already recognised that the standard leaves too much room for interpretation in the structuring and use of the concepts and their instances. The new version makes the application of the standard clearer.

As the older standard ISO 12006-3:2016 provides the metamodel for our property server datacat, we analysed the changes resulting from the new version. Two major changes have a significant impact on the use of the metamodel for creating concepts. These concern the grouping of concepts and the relationships between concepts. There are also many smaller changes that do not have a major impact on the metamodel.

Previously, the concepts *xtdCollection* and its sub-concepts *xtdNest* and *xtdBag* were used for grouping concepts. These concepts do not exist in the new version. Groupings of objects of the same concept type (*xtdNest*) can be used as objects of the concepts *xtdSubject* or *xtdProperty* with the attributes *ConnectedSubjects* or *ConnectedProperties* and relationships to the objects in the group. Figure 4 shows an example of the grouping of features in both metamodels. The grouping of objects of different types (*xtdBag*) can no longer be displayed and must be resolved differently when mapping data dictionaries.

The concept of relationships has been completely revised. Instead of the abstract concept *xtdRelationship* with its many specialised relationship classes like *xtdRelCollects*, *xtdRelAssignsValues* and *xtdRelSpecializes*, there are now only two generic objectified relationships *xtdRelationshipToSubject* and *xtdRelationshipToProperty* (Figure 5). In particular, this allows the *xtdRelCollects* relationships to be replaced. Relationships previously used as objects such as *xtdRelAssignsValues* are replaced by simple properties such as *Values*.

In addition to these two significant changes for us, the sub-concepts *xtdActor*, *xtdActivity* and *xtdMeasureWithUnit* of *xtdObject* have also been removed. Instead, the new version has, for example,

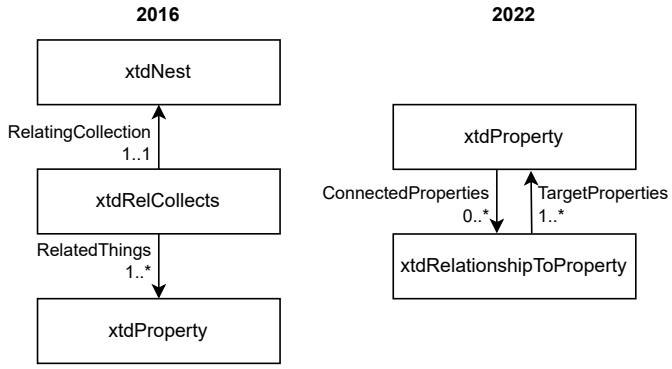


**Figure 4:** Collection of properties in the versions of ISO 12006-3:2016 and ISO 12006-3:2022

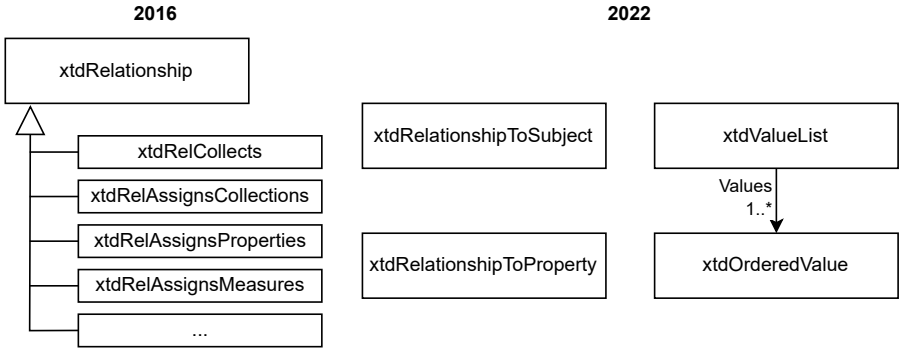


**Figure 5:** Relationships in the versions of ISO 12006-3:2016 and ISO 12006-3:2022

the concepts *xtdConcept* and *xtdCountry*. The inheritance hierarchy between concepts has also been changed. The abstract concept *xtdObject* is inherited by other concepts under *xtdRoot*, including the abstract concept *xtdConcept*. Most of the concepts that are important to us inherit from this concept. Not only the hierarchy, but also the attributes of the existing concepts have been changed and many new attributes have been added, such as *DataFormat*, *DataType* and *BoundaryValues* for *xtdProperty*.

The result of the analysis is that a large part of the metamodel in the existing software needs to be adapted and a migration of existing technical concepts and data dictionaries needs to be carried out. The changes are currently being implemented both in the backend of our property server datacat and in the OWL export. In addition, migration processes are being developed for existing data dictionaries at the Neo4j graph database level so that they can continue to be used. The migration will be implemented using Cypher queries, which will be used to create new links, delete redundant relationship classes, and add concepts and attributes.

However, migration from ISO 12006-3:2016 to ISO 12006-3:2022 is not fully possible for all existing concepts. For example, grouping of different concept types is no longer possible. In addition, values can no longer be attached to other concepts with a descriptive property via the *xtdRelAssignsPropertyWithValues* relationship. Properties can only be attached to *xtdSubject* concepts. Values must be part of a value list attached to the property. In these cases, design decisions must be made to preserve as much information as possible during migration.

## 5. Results and Limitations

The approaches developed so far for exporting data dictionaries and concepts from property servers with semantic meaning have shown that Semantic Web technologies can be used in a variety of ways. Since the meta concept standard ISO 12006-3 does not provide a schema nor implementation rules,

there is no single solution for creating ontologies and describing data in RDF. We have decided to use existing, established standards wherever possible to create a structured, standardised ontology export. Our aim is to make ontologies comparable and to find possible ways of linking them. The ontologies of neighbouring domains can be searched for the same or similar terms for linking. However, it is not automatically the case that these terms are used in the same way. But, using the ISO 12006-3 metamodel, the ontologies contain a conceptual schema that indicates the level of meaning at which each term is used. For example, if the term *Lantern* is described conceptually as *XtdSubject* in both ontologies, it can be assumed that it is used very similarly or identically in both domains and is therefore suitable for linking. However, if the term is described once as *XtdSubject* and once as *XtdValue*, the usage is different and maybe should not be used for linking. With this approach, we are creating a technical and methodological basis for our research into the shared and cross-domain use of data dictionaries in BIM and GIS. Using our existing property server datacat as a starting point, we were able to successfully demonstrate an ontology export. Entire data dictionaries or individual concepts can be output as ontologies in valid Turtle syntax. The reference to the data dictionary as well as the concepts and the relationships between them are retained, although the relationships are simplified. However, the presented export is only a proof of concept to test the methodology. The ontologies can be published via persistent URIs once the modelling of the data dictionary in the property server has been completed. The tested ontology has not yet been published, as the content of the property server is still being developed by the expert group and the licensing conditions have not yet been clarified.

There are still some limitations to our approach. The biggest limitation is the use of the old ISO 12006-3 standard from 2016. Now that we have familiarised ourselves with the new version of the ISO 12006-3:2022 standard, we want to implement it in our datacat software and subsequently also in the ontology export. A comparison of the 2016 and 2022 versions has shown that there are some significant changes in the described metamodel. In our view, the new 2022 version provides a more clearly defined, more generic metamodel that can be used to better describe data dictionaries. The ontology export needs to be adapted at a technical level and the handling of values and units needs to be reconsidered.

Figure 6 shows how the concepts in the layers change compared to Figure 2 when the new metamodel is implemented. Changes in the implementation due to the new metamodel are underlined. In Concept Dictionary Layer, fewer instances need to be created for relationships because attributes are used for a direct relationship between concepts. The link between value list and value needs to be ordered in the new metamodel. Therefore, a new class *OrderedValue* is added in the new Ontology Layer, which receives an attribute *orderNumber*. This allows the value to be reused in other lists with a different order number. The *owl:ObjectProperty hasOrderedValue* and *hasValue* are used to create the relationships. It is important to us that value lists and possible values are also included in the ontology, as this means that there is no need to refer to external lists that restrict the value range of the characteristics. In some cases, the order of the values is also important, so we follow the standard definition here.

During the update process, it is important to be aware of the limitations of the lack of standardisation of units and values in datacat. As the new ISO 12006-3:2022 standard is not clear on the use of units, it is necessary to use collections with predefined units that allow the use of standardised units to ensure clarity. This means that users will make fewer mistakes and always refer to units in the same way, which will also improve the quality of the data dictionaries and ontologies.

## 6. Conclusion and Outlook

With our approach, we have shown that the ontology export of data dictionaries or concepts from a property server based on the ISO 12006-3 standard can be implemented very closely to standards with existing ontologies. The Dublin Core ensures that the metadata from the property server is not lost when the ontologies are exported. In addition, the metadata can be used to draw clear conclusions about the concepts of the data dictionary from the ontology elements. The ISO 12006-3 metamodel is present in the ontology without the users of the ontology having to know and understand it, thus ensuring a standardised structure when comparing ontologies that have been created. The ontology receives all

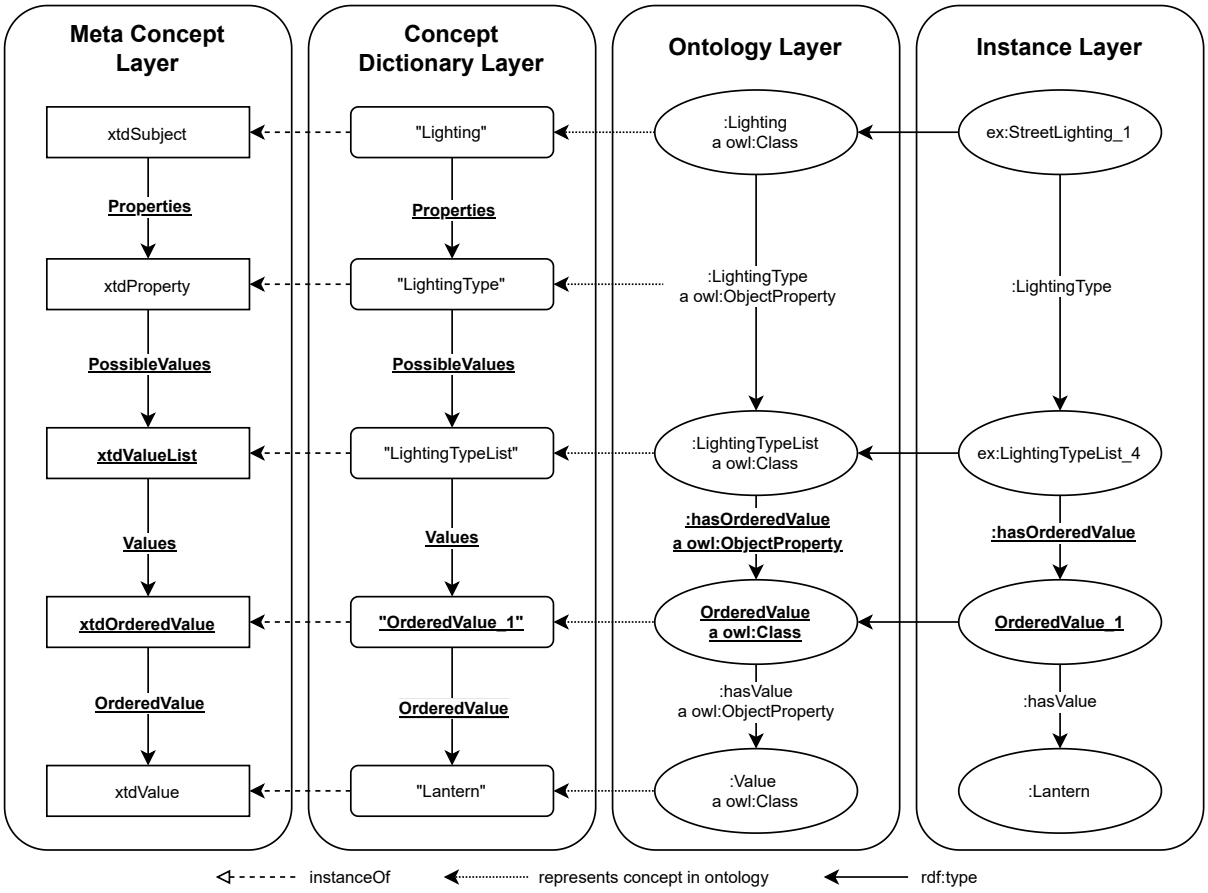


**Figure 6:** Concepts after updating the Meta Concept Layer (based on [8]). The underlined parts have changed.

information from the property server, although some relationships are simplified so that the ontology does not become unnecessarily complex. Unlike the bSDD, our property server datacat can output all dictionary elements, whereas the bSDD only outputs instances in RDF, while we write an ontology. As with the other approaches presented, our method is not entirely free of design choices. The open approach of Semantic Web technologies allows for many variants. The advantage of our variant is that the ontology is closely linked to standards, while the disadvantage is that we had to introduce our own concepts for linking concepts during the ontology export.

However, in order to further reduce the disadvantages and limitations, the work has also provided new ideas for the further development of the approach. Initially, the focus will be on updating the metamodel in the backend of datacat and adapting the frontend to the new requirements. The next step is to integrate the exported ontologies in other software for the practical usage.

Ontologies are not used much in daily practice, neither in BIM nor in GIS. Our research aims to develop common data dictionaries and ontologies and link existing ones to improve the integration of BIM and GIS data and to make the data dictionaries usable across domains. We suppose, that ontologies from different domains, and thus the data dictionaries behind them, can be more easily linked if they are structured according to the same standardised meta schema. The research approaches presented in this paper are intended to provide the technical basis for this.

## Acknowledgments

This work is co-funded by the European Union and the Free State of Saxony as part of the ESF Plus programme (Funding Number: 100670485).

Co-funded by the European Union

This project is co-financed from tax revenues on the basis of the budget adopted by the Saxon State Parliament.

## References

[1] International Organization for Standardization, Building construction - Organization of information about construction works: Part 3: Framework for object-oriented information, Technical Report 12006-3, 2016. URL: https://www.iso.org/standard/38706.html.

[2] International Organization for Standardization, Geographic information – Reference model: Part 1: Fundamentals, Technical Report 19101-1, 2014. URL: https://www.iso.org/standard/59164.html.

[3] International Organization for Standardization, Geographic information – Methodology for feature cataloguing, Technical Report 19110, 2016. URL: https://www.beuth.de/de/norm/din-en-iso-19110/267630806.

[4] G. Fröch, W. Gächter, A. Tautschnig, G. Specht, Merkmalserver im open–bim–prozess, Bautechnik 96 (2019) pp. 338–347. doi:10.1002/bate.201800092.

[5] J. Oraskari, Live web ontology for buildingsmart data dictionary, 2021. URL: https://www.researchgate.net/publication/355425683_Live_Web_Ontology_for_buildingSMART_Data_Dictionary#fullTextFileContent.

[6] C. Zhang, J. Beetz, B. de Vries, An ontological approach for semantic validation of ifc models, in: Proceedings of the 21st International Workshop on Intelligent Computing in Engineering, Cardiff, United Kingdom, Curran Associates, Inc., Red Hook, 2014, pp. 1–8. URL: https://www.researchgate.net/publication/266326240_An_Ontological_Approach_for_Semantic_Validation_of_IFC_Models.

[7] P. Pauwels, T. Krijnen, J. Beetz, Making sense of building data and building product data, 2016. URL: http://babelnet.org/lux/files/4.%20pauwels%20et%20al.%20-%20making%20sense%20of%20building%20data%20and%20building%20product%20data.pdf.

[8] J. Beetz, B. de Vries, Building product catalogues on the semantic web, in: A. Dikbas, E. ergen, H. Giritli (Eds.), Proceedings of the 26th International Conference on IT in Construction & 1st International Conference on Managing Construction for Tomorow, Istanbul, Turkey, CRC Press and Balkema, Leiden, 2009, pp. 221–226. URL: https://www.researchgate.net/publication/260707450_Building_product_catalogues_on_the_semantic_web.

[9] Y. Yao, R. Wu, H. Liu, Jtowl: A json to owl converto, in: Y. Zeng, S. Kotoulas, Z. Huang (Eds.), Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning, Shanghai, China, ACM, New York, NY, USA, 2014, pp. 13–14. doi:10.1145/2663792.2663801.

[10] A. Wagner, U. Rüppel, Bpo: The building product ontology for assembled products, in: Proceedings of the 7th Linked Data in Architecture and Construction Workshop - LDAC2019, Lisbon, Portugal, 2019, pp. 106–119. URL: http://tubiblio.ulb.tu-darmstadt.de/115951/.

[11] M. Bonduel, P. Pauwels, R. Klein, Property modelling in the aeco industry, in: P. Pauwels, K. McGlinn (Eds.), Buildings and Semantics, CRC Press, London, 2022, pp. 25–50. doi:10.1201/9781003204381-3.

[12] C. Clemen, B. Thurm, S. Schilling, Managing and publishing standardized data catalogues to support bim processes, in: Proceedings of the 38th International Conference of CIB W78, Luxembourg, 2021. URL: https://www.researchgate.net/publication/355486637_Managing_and_publishing_standardized_data_catalogues_to_support_BIM_processes.

[13] International Organization for Standardization, Building information modelling (BIM) Data templates for construction objects used in the life cycle of built assets: Concepts and principles,

Technical Report 23387, 2020. URL: https://www.iso.org/standard/75403.html.
[14] International Organization for Standardization, Building construction - Organization of information about construction works: Part 3: Framework for object-oriented information, Technical Report 12006-3, 2022. URL: https://www.iso.org/standard/74932.html.