2019 European Conference on Computing in Construction
Chania, Crete, Greece
July 10-12, 2019

RELATING GEOMETRY DESCRIPTIONS TO ITS DERIVATIVES ON THE WEB

Anna Wagner[1]*, Mathias Bonduel[2], Pieter Pauwels[3] and Uwe Rüppel[1]
[1]Department of Civil and Environmental Engineering Sciences, TU Darmstadt, Darmstadt, Germany
[2]Department of Civil Engineering, Technology Cluster Construction, KU Leuven, Ghent, Belgium
[3]Department of Architecture and Urban Planning, Ghent University, Ghent, Belgium

Abstract

Sharing building information over the Web is becoming more popular, leading to advances in describing building models in a Semantic Web context. However, those descriptions lack unified approaches for linking geometry descriptions to building elements, derived properties and derived other geometry descriptions. To bridge this gap, we analyse the basic characteristics of geometric dependencies and propose the Ontology for Managing Geometry (OMG) based on this analysis. In this paper, we present our results and show how the OMG provides means to link geometric and non-geometric data in meaningful ways. Thus, exchanging building data, including geometry, on the Web becomes more efficient.

Introduction

The exchange of building data in a Semantic Web context is of growing interest for researches related to the construction industry. However, current results mainly focus on non-geometric or topological building data, as can be seen in Pauwels et al. (2017). The description of geometry on the Web is lagging behind compared to the developments in other fields of interest to the construction industry, hindering the exchange of complete building data using Semantic Web technologies.

To our knowledge, conducted research on the topic of geometry representation in a Semantic Web context is lacking unified and meaningful connections for geometry descriptions. This affects the relation between (building) elements and their geometry description as well as dependencies between geometry descriptions and properties derived from it (e.g. volumes, dimensions, and areas) or other derived geometric representations (e.g. a bounding box derived from a chair geometry). Nonetheless, these dependencies are needed on every-day basis to ensure data integrity during modelling, exchange and collaboration processes and have already been discussed for geometry descriptions outside of the Semantic Web domain (Mirtschin, Jon 2018) as well as inside of it (Zhang et al. 2017). This shows the need for an approach to enable the creation of such substantial connections between geometry derivatives and their origins within a Semantic Web context.

To fulfil this requirement, this paper aims to serve as a foundation for further research to ensure the integrity of geometric data on the Web and therefore summarises different use cases that may occur in this background while also proposing an approach on how to model the observed relations. Our approach is based on an analysis of necessary relations between geometry descriptions and their derivatives to create a meaningful vocabulary to express them. Further, we investigate the publicly available Ontology for Property Management or OPM[1] (Rasmussen, Lefrançois, Bonduel, Hviid & Karlshøj 2018) regarding its applicability towards geometry descriptions, as we understand geometry description to be a specific kind of property of an object.

Next, we conduct a literature review of existing approaches for connecting different geometry representations and related fields such as property management. This is followed by an analysis of the three different types of geometric dependencies we identified: 1) between an object and its geometry, 2) between geometries that describe the same object, and 3) between non-geometric properties and geometry. With these insights in mind, we introduce the Ontology for Managing Geometry (OMG) in detail, after which we review the OMG using example data and competency questions deduced from the previously presented analysis. Finally, we conclude this paper with a discussion of our results and give a brief outlook on the next necessary steps to ensure parametric modelling on the Web.

Related Work

The conducted literature review includes four topics that align with the identified types of geometric dependencies: First, the application of multiple geometry descriptions in the built environment. Second, ensuring data integrity while handling geometry descriptions of different geometry representations. Third, modelling of parametric coherence between geometry descriptions and non-geometric data, and fourth, the representation of data that changes over time.

Beginning with the application of multiple geometry de-

*Corresponding author: wagner@iib.tu-darmstadt.de

[1]https://w3id.org/opm

DOI:10.35490/EC3.2019.146

scriptions, we found that this is – especially in the context of Level of Detail or Level of Geometry – a highly anticipated topic. For example, the CityGML and IFC standards already support multiple geometry descriptions to describe environments and buildings in different levels of detail (Ohori et al. (2015), IFC). The definition of the geometry representations, however, is predefined within these standards. Thus, users are limited in their choice of geometry representations. Especially for as-built BIM this can be a disadvantage, since the as-built geometry is usually captured in point clouds, that are currently not supported by the IFC standard. Therefore, Krijnen & Beetz (2017) propose a point cloud extension of the IFC schema to allow the collection of all building data within one file. While the extension of EXPRESS-based standards seems cumbersome, such an extension could easily be achieved using Semantic Web technologies.

Still, when multiple geometry descriptions are provided for the same object, the risk of data discrepancies arises unless specific routines are set up to ensure data integrity. One option to do so is presented in Pauwels et al. (2011), where the authors propose a methodology to convert geometry descriptions of different geometry formats using a Semantic Web interface. Yet, the authors suggest to create this conversion on the fly instead of storing both geometry descriptions persistently. This may be suitable if the second geometry description is rarely needed and serves for archiving or exchange purposes only. In the latter case, this would still hold the disadvantage that the link between the original geometry description and the converted one might get lost during the exchange. For these reasons, we suggest to keep all geometry descriptions attached to the object, though algorithms and methodologies as the one presented by Pauwels et al. (2011) could be applied for keeping the data consistent.

When it comes to keeping the data consistent, one must also consider the relations between non-geometric and geometric data, as those are commonly dependent from each other. Correspondingly, data integrity is already an issue for building models that only contain one geometry description, e.g. when the height of an element is also described as a non-geometric property. To ensure data integrity, these dependencies must be described in detail to update the data accordingly. In a Semantic Web context, Zhang et al. (2017) introduce BimSPARQL to automatically retrieve a selection of properties that can be derived from the object's geometry description. They propose a new vocabulary with underlying SPARQL functions to enhance querying building data. However, this approach only takes specific properties into account and must be extended for the introduction of new dependencies. Another approach was presented by Wagner et al. (2018), in which the authors present a product ontology including parametric coherences between geometry descriptions and non-geometric properties. With this approach, any relation can be modelled using an ontology dedicated to describe equations and mathematical operations. Nonetheless, the described equations need to be translated and executed by a math kernel to generate the desired output. Both approaches allow additional relations between non-geometric and geometric data, but are lacking methods to easily identify outdated data and thus must be executed every time the data is queried. By storing the calculated results within the data and keeping it synchronised, handling (parametric) dependencies would become simpler and more effective. Apart from the efforts made in the construction industry, ontologies from different domains – such as the PROV ontology[2] – introduce concepts to describe such derivations persistently.

In order to keep dependent data synchronised, it must be possible to model data that evolves over time. While it has been an objective in research to create methodologies to keep track of the evolution of ontologies in general, these approaches (e.g. Pittet et al. (2014)) focus on the ontology's schema level (TBox) and can therefore not directly be applied to instance level (ABox). Within the W3C Linked Building Data Community Group (W3C LBD CG)[3], efforts have been made to allow version control of properties. The basic idea of the group was to allow the modelling of properties on three levels with different options for adding metadata (Bonduel 2018). On the first level, the property is directly connected to the building object using individual datatype properties for effective querying. Additional information, e.g. the used unit, could be described using custom datatypes (Lefrançois & Zimmermann 2016). The second level introduces an intermediate node for the property which contains the property's value and can also hold additional metadata, e.g. the used unit, the author, or the timestamp of the property's creation respectively last update. This already allows the inclusion of most relevant metadata, but can still only hold one value for the property, thereby losing older value entries when updating. The third and highest level also enables a persistent recording of the property's history. Between the property node and the property's value, another node for the property state is inserted. Thus, one property can have multiple states – and with that values – while during queries only the current state would be retrieved unless the user defined otherwise. This third level for property modelling can be realised using the Ontology for Property Management (OPM, Rasmussen, Lefrançois, Bonduel, Hviid & Karlshøj (2018)).

The OPM and three levels approach of the W3C LBD CG provide methods to address some of the core features

[2]http://www.w3.org/ns/prov
[3]https://w3c-lbd-cg.github.io/lbd/

desired for relating geometry descriptions to their derivatives, namely a stable, modular and efficient way for adding metadata to properties and keeping track of their change history. Since we understand geometry to be closely related to properties, the work presented in this paper is oriented towards these approaches and will also use multiple levels for modelling geometry. At the same time, the OPM itself cannot be reused as RDF-based geometry descriptions are not just based on properties but also on concepts (e.g. box, sphere, and so forth). Even non-RDF-based geometry descriptions may depend on multiple files, i.e. OBJ geometries that can be enriched by material descriptions which must be stored in a separate file. Hence, we need to identify all relevant relations that should be depicted to create a new ontology to address all of these collected requirements.

Geometric dependencies

In our analysis, we identified three types of geometric dependencies: (1) the connection between an object and its geometric representation(s), (2) semantically meaningful dependencies between geometry representations of the same object, and (3) relations between geometries and their derived properties. Following, our findings on these dependencies will be explained in more detail.

Geometry descriptions and objects

Discussed dependencies of this paragraph occur between geometry descriptions and the objects they are portraying. For connecting geometry descriptions and objects, no restriction towards the relation's cardinality should be imposed on the user to allow multiple geometry descriptions. Howbeit, if multiple geometry representations are connected to one object, these representations should be enriched with metadata to enable meaningful differentiation of them. Ergo, a flexible and modular approach to connect geometry descriptions and objects is needed that enables the addition of different amounts of metadata in respectively of the current use case. This could be realised by introducing different levels of this relation, similar as suggested by the W3C LBD CG for property modelling (Bonduel 2018).

At the same time, the connection should be generalised for RDF-based and non-RDF-based geometry descriptions to ease unified querying for geometry. In this regard, the introduction of a geometry context as metadata would further simplify the extraction of a specific kind of geometry model from the entire graph (e.g. planner models).

Moreover, it should be possible – while still optional – to add geometry states to the geometry description for version control purposes. Similar to how version control can be realised by the OPM's property states, geometry states could be introduced. But with RDF-based geometry descriptions in mind, it does not seem feasible to create a new geometry state for every change on properties within the geometry description. Instead, the OPM could be used to describe such property changes, while changes on (object) node level (e.g. class types, deletion of nodes) must be recorded by the geometry state.

Geometry descriptions of the same object

This category covers dependencies between geometry descriptions only. Relations between geometry descriptions of the same object mainly focus on the definition of which description can be derived from another. In some cases, this relation may be bidirectional, allowing to convert the geometry descriptions back and forth. Additionally, when version control is wanted, this relation should also be manifested on geometry state level, defining the conducted derivation of the specific geometry states. Without relations on state level, unnecessary derivations or even endless loops of deriving geometry descriptions from each other may occur, as they are derived based only on timestamps. Also, geometry descriptions that cannot be used to (indirectly) derive any other geometry representation of the same object, should be classified as read-only to prevent changes that cannot be deduced back into the data pool.

Apart from relations to describe derivabilities, geometry descriptions can also be dependent on each other in different ways. For example, a geometry description could serve as a supplement for another one, as is often the case in heritage models. The underlying geometry description is stored in a simplified mesh or solid geometry, while details are represented in point clouds. It is not always feasible, wanted or even possible, to convert fine details to other geometry representations, so it should be possible to model some kind of relation where one geometry description supplements another.

Another type of relation would be the reusage of previously defined geometry descriptions. This may happen in one of two ways: Either the first geometry description is reused without changes, e.g. when multiple instances of one object type such as a particular kind of chair are placed within a model, or when the first geometry is used as a basis for further development, while the original geometry is still needed as it was modelled before. An example for this would be a bounding box created in the early planning stages to induce the idea of a chair within a room which is subsequently modelled in more detail based on the original bounding box. Since the bounding box representation may still be of relevance for simulation use cases, it should not be overwritten. In both cases, the relation is purely informative and does not contain any further details on possible derivations. If such dependencies are required, they have to be modelled separately.

Geometry descriptions and derived properties

The last category of geometric dependencies is between geometry descriptions and their derived properties. Such derivations may be implicit or explicit. As an example, the height of a wall may be explicitly defined in the wall's geometry description (e.g. if it is an extrusion), while it is commonly also described as a non-geometric property. To avoid redundant and conflicting data, the non-geometric property should be linked to the geometric property.

Implicitly derived properties, on the other hand, can not be derived from one singular geometry property, but require mathematical evaluation of the geometry (e.g. the volume of the wall, which is dependent on the base area, extrusion and possibly even voids within the wall). To emphasise this relation, the property should be connected to the geometry description itself. However, the calculation of the new property value after changing the geometry, must be calculated externally.

Ontology for Managing Geometry (OMG)

Based on the previous analysis and the concepts and methods presented in the OPM ontology, the Ontology for Geometry Management (OMG)[4] was developed. The ontology is also publicly available on GitLab [5].

The OMG ontology aims to provide means to connect geometry descriptions to building data and relate geometries with derived or dependent geometries and properties. To create a more consise ontology, we defined that the exact description of parametric dependencies, algorithms to derive geometry representations from each other, and the geometry description itself are out of scope of this ontology.

Within the OMG, several concepts from other ontologies were reused. Furthermore, the example data contains additional schemata, e.g. for geometry descriptions. To ease the understanding of the upcoming figures and examples, Listing 1 introduces all used ontologies and prefixes in this paper.

```
@prefix omg: <https://w3id.org/omg#> .
@prefix fog: <https://w3id.org/fog#> .
@prefix opm: <https://w3id.org/opm#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix seas: <https://w3id.org/seas/> .
@prefix schema: <http://schema.org/> .
@prefix bot: <https://w3id.org/bot#> .
@prefix geom: <http://rdf.bg/geometry.ttl#> .
```

Listing 1: Used ontologies and prefixes.

The OMG enables – similar to the proposed property modelling by the W3C LBD CG (Bonduel 2018) – the modelling of relations between objects and geometry on three levels. While the first level provides means to directly connect geometry descriptions to objects, the levels 2 and 3 introduce additional intermediate geometry and geometry state nodes to allow metadata to be attached to the geometry. Within the following figures of the OMG architecture, objects, properties and classes that are not part of the presented OMG ontology are greyed out for better understanding of the new concepts.

Figure 1 depicts how geometry can be connected to an object using the OMG level 1. The two proposed properties `omg:hasComplexGeometryDescription` and `omg:hasSimpleGeometryDescription` can be used to link to the first node of an RDF-based geometry description or to add non-RDF geometry descriptions as literals or paths to external files. Since these relations are generic and hold no information about the used geometry format, individual relations should be introduced as subproperties for used geometry formats and their versions. For this purpose, the File Ontology for Geometry formats (FOG)[6] ontology was created (Bonduel et al. 2019) and can be further extended in its openly available GitHub repository[7].

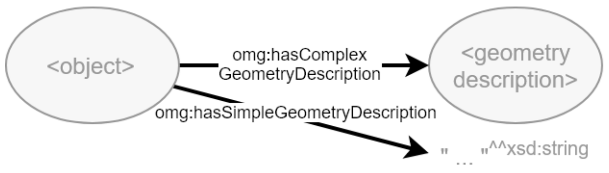
<object>
omg:hasComplex GeometryDescription
<geometry description>
omg:hasSimpleGeometryDescription
"..."^^xsd:string

Figure 1: OMG level 1.

If more metadata needs to be added to the graph, the OMG level 2 includes the `omg:Geometry` and `omg:GeometryContext` classes. A geometry node can be connected to a geometry context and other geometry nodes (dependencies), and can be used to enrich the geometry description with metadata as the author or timestamp of creation using other ontologies as the PROV ontology or schema.org. The geometry description itself can then be attached to the geometry node with the same properties as those introduced for OMG level 1. A potential layout of a graph with a OMG level 2 description can be seen in Fig. 2.

Any object can be connected to one or multiple geometry nodes using the `omg:hasGeometry` relation. If multiple geometry nodes are applied, it is possible to relate them to each other to describe their dependencies. For instance, geometry nodes can be linked using `omg:isDerivedFromGeometry` to indicate that the first geometry has been derived from the second.

For cases where one geometry complements another, as is often the case with point clouds and tessellated ge-

[4]https://w3id.org/omg

[5]https://gitlab.iib.tu-darmstadt.de/Wagner/omg.git

[6]https://w3id.org/fog

[7]https://github.com/mathib/fog-ontology

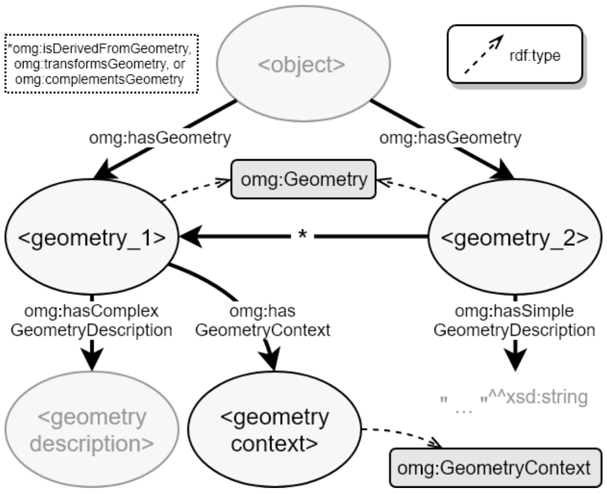

Figure 2: OMG level 2.

ometries, the opm:complementsGeometry relation can be used. However, note that this relation does not imply any dependencies on the geometries' derivations. Additionally, the omg:transformsGeometry relation can connect geometries that share the same base geometry but are placed in different locations. This might be the case for geometries of the same object, where one geometry representation is placed in its local coordinate system and other geometries reuse the original geometry and place it in different global or other local coordinate systems. Another example would be multiple instances of the same geometry like doors or windows. If this relation is used, the connecting geometry node should only contain information about the transformation, e.g. a transformation matrix, while the geometry description itself should be defined in the connected, original geometry node. Finally, a geometry context can be related to a geometry node with the omg:hasGeometryContext property. By using a geometry context, the extraction of multiple geometry descriptions from an entire graph that are relevant for a specific use case can be simplified.

The last level also includes concepts to manage evolving geometry descriptions. An overview of OMG level 3 is shown in Fig. 3. It introduces the new class omg:GeometryState, which is closely related to the opm:PropertyState and can also be connected to a geometry context. Inspired by the idea of the opm:CurrentPropertyState, the OMG also introduces a subclass for the geometry state to define those states that are currently valid: omg:CurrentGeometryState.

A geometry state describes an object's geometry at a specific point in time, which poses the requirement of adding a timestamp to each geometry state. Using the omg:hasGeometryState relation, one or more geome-

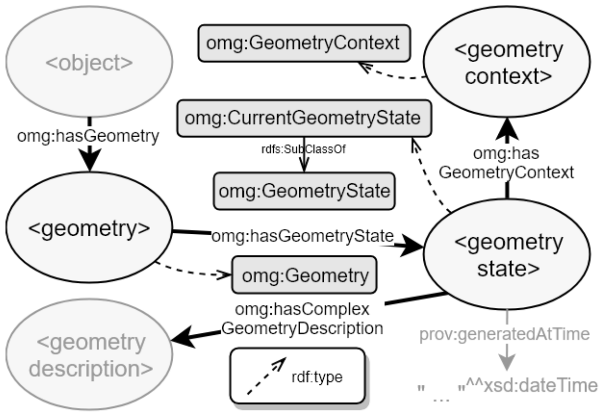

Figure 3: OMG level 3 – overview.

try states can be related to a geometry node. When OMG level 3 is applied, the geometry descriptions are connected to the geometry states. As soon as the geometry description is changed, a new geometry state with the changed description should be added. However, since it is not feasible to create a new geometry node with a copy of widely unchanged RDF-based geometry descriptions when only one property of the description changes, the OMG ontology should be combined with the OPM ontology for RDF-based geometry descriptions. Thus, geometry states can be connected to any geometry object within the geometry description using the omg:containsGeometryObject relation. Subsequently, the chain property omg:containsPropertyState will infer a link between any property states of the geometry description and the geometry state itself based on the former relation and the OPM architecture (see Fig. 4).

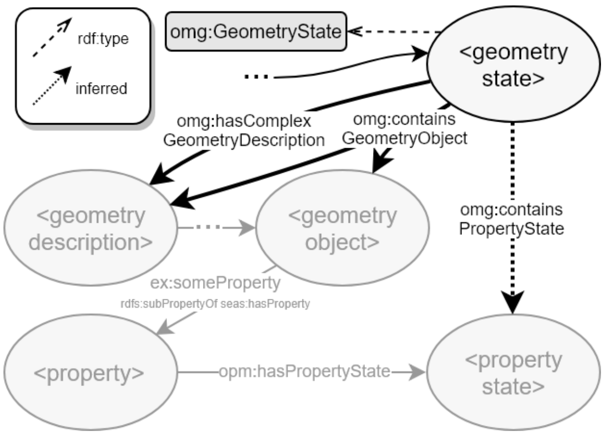

Figure 4: OMG level 3 – geometry states.

Apart of keeping track of current changes, geometry states can also be used to enhance the definition of geometry derivations as was introduced in OMG level 2.

As shown in Fig. 5, this definition takes place on two layers: between geometry nodes and between geometry state nodes. In contrast to the application of the omg:isDerivedFromGeometry property for level 2, this property is now used to indicate the *possibility* to derive a geometry. Thus, one geometry node may have multiple other geometry nodes connected this way and in case of bidirectional dependencies, the property can be added twice between two nodes, pointing in different directions. To define the specific derivation of one geometry state, the omg:isDerivedFromGeometryState property is introduced.

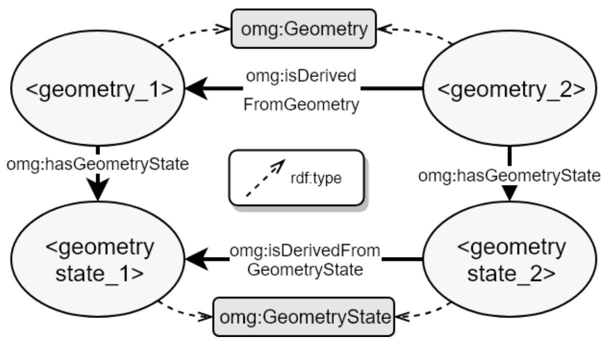

Figure 5: OMG level 3 – derived geometry states.

To correctly model these dependencies, any modified (changes on property level of an RDF-based geometry description) or new geometry state (changes on object level of an RDF-based geometry description or changes in non-RDF-based geometry descriptions) should be modelled without the omg:isDerivedFromGeometryState property first. This means for the case of modified geometry states, that this property must be removed. Afterwards, geometry states must be connected to a description's geometry state using the omg:isDerivedFromGeometryState property if they were modified or created by derivation of that geometry description. With this approach, infinite loops while updating geometry descriptions can be prevented: Since newly generated geometry descriptions will be stored in a new state, this state's timestamp will be more current than the one of the initially changed geometry state. If the dependency is modelled bidirectionally, this may cause a loop. Thus, the new relation was defined to pinpoint the origin of a new geometry state and will ensure that the new state will not cause its origin's update. If the state was created because of manual changes, this relation will not be used and thereby the data discrepancy can be detected.

Independent of the OMG levels, relations dedicated to describe dependencies between geometries and properties are defined. As mentioned before, we differentiate between explicit and implicit dependencies. Figure 6 depicts the introduced relations for the first case. The omg:isExplicitlyDerivedFrom property connects two seas:Property nodes – one of the semantic description of the object and one of the geometric description. Based on this relation and the architecture of the OPM, the chain property omg:hasInferredPropertyState will infer the relation between the derived property and any opm:PropertyState of its origin property.

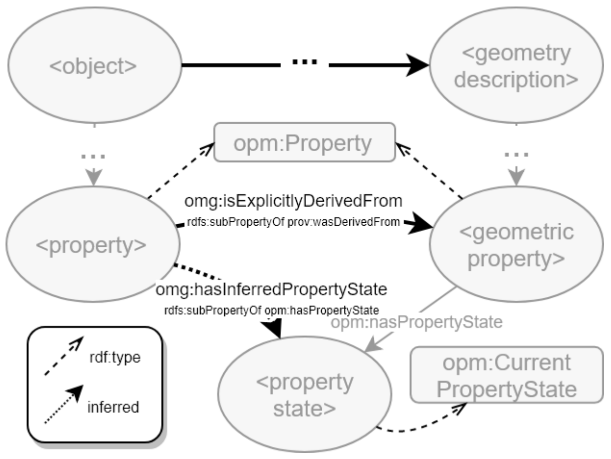
<object>
...
<geometry description>
...
opm:Property
...
<property>
omg:isExplicitlyDerivedFrom
rdfs:subPropertyOf prov:wasDerivedFrom
<geometric property>
omg:hasInferredPropertyState
rdfs:subPropertyOf opm:hasPropertyState
opm:hasPropertyState
rdf:type
inferred
<property state>
opm:Current PropertyState

Figure 6: Explicit property derivation.

The relation for implicit dependencies is simpler, as no values or states can directly be inferred. Instead, the omg:isImplicitlyDerivedFrom property connects any property to a geometry node (see Fig. 7).

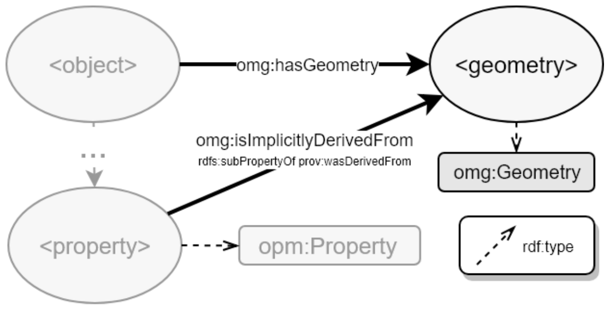

Figure 7: Implicit property derivation.

Recommendations for using OMG

When applying the OMG, we recommend to first evaluate the requirements of the considered (part of the) project before defining which level will be used to avoid unnecessary overhead or too rigid data structures. The functionalities of the different levels are also shown in Table 1.

Note that it is in principle possible to attach multiple geometries to a building element using OMG level 1. However, since it is not possible to define derivations between or geometry contexts of geometry descriptions in general,

Table 1: Functionalities of the OMG levels.

Functionality	level 1	level 2	level 3
Connecting geometry	yes	yes	yes
Multiple geometries	no*	yes	yes
Defining derivations	no	yes	yes
Defining context(s)	no	yes	yes
Version control	no	no	yes
Explicit properties	yes	yes	yes
Implicit properties	no	yes	yes

and – even worse – to describe differences between multiple geometry descriptions of the same format, we highly recommend not to do so. Additionally, when non-RDF-based geometry descriptions that require multiple literals are attached directly to the object, it might be problematic to identify which literals belong together, especially when multiple geometry representations are attached using OMG level 1.

For version control on complex, RDF-based geometry descriptions (OMG level 3), we also highly encourage the combination of the OMG with the OPM to prevent unnecessary duplicates of geometry descriptions when only one value changed.

Alignment of OMG

Since the OMG was designed as a generic ontology to attach geometry to any object, it can be used in any field of application in principle. However, as we originally designed it to support the application of linked building data, we propose alignments to mostly building specific ontologies in this section.

The biggest overlap of the OMG is with the OPM. The concepts of both ontologies are strongly related, however, since geometry cannot be classified as a subtype of properties, it did not seem feasible to extend the OPM ontology. Thus, we suggest to create an upper level management ontology to pool both OPM and OMG with their concepts – especially the property and geometry states – together. This would enable concepts of the OPM that are also applicable to the OMG, e.g. `opm:Deleted` or `opm:Assumed`, to be transferred to the upper level ontology and thus be used by both, OPM and OMG. Also, the introduced `omg:hasInferredPropertyState` property should be transferred to the OPM, since it is mainly using concepts, classes and relations of the OPM.

Another close relative of the OMG is the File Ontology for Geometry formats (FOG), that extends the OMG to create file format and ontology specific subproperties of the `omg:hasComplexGeometryDescription` and `omg:hasSimpleGeometryDescription` properties.

In the background of the built environment, another important alignment is towards the Building Topology Ontology (BOT). The BOT already introduces properties to describe RDF- and non-RDF-based geometries to building elements or zones: `bot:hasSimple3DModel` and `bot:has3DModel` (Rasmussen, Frausing, Hviid & Karlshøj 2018). However, since these relations confine to 3D geometry, while the OMG can also be used to connect 2D geometries, we refrained in reusing those properties. Instead, we suggest to rename the properties to `bot:hasComplexModel` and `bot:hasSimpleModel` and align them with the OMG properties `omg:hasComplexGeometryDescription` and `omg:hasSimpleGeometryDescription`. While aligning these properties, one should consider the consequences of `owl:equivalentProperty` regarding the property's range and domain restrictions. Another option would be to remove these properties entirely from the BOT ontology and instead use the concepts introduced in OMG.

Furthermore, all properties that indicate (parametric) dependencies of geometries and other geometries or properties are defined as subproperties of the `prov:wasDerivedFrom` property of the PROV ontology.

Proof of concept

To prove the functionalities of the OMG ontology, an openly available *SPARQL-visualizer* demo was created to address the underlying competency questions of the ontology[8]. The demo provides example data that can be used for all of its steps. However, for the more complex steps, i.e. manipulating the database or querying with reasoning, a Stardog triplestore must be connected to the visualiser. The competency questions (CQ) that are answered by the demo are explained next.

CQ1: How can a single geometry description be connected to an object directly?

This question is used to demonstrate the simplest functionalities of OMG as might be of use for data exchange. It can be fulfilled by using OMG level 1 as shown in Fig. 1 and serves as the basis for all other competency questions and functionalities.

CQ2: How can multiple geometry descriptions with their dependencies be related to an object?

In case multiple geometry descriptions are required, they must be attached to an object in meaningful ways, allowing the expression of the descriptions' dependencies. Based on this question, OMG level 2 was developed and therefore this level can address it (see Fig. 2).

[8]https://madsholten.github.io/sparql-visualizer/?file=https:%2F%2Fwww.dropbox.com%2Fs%2Fg1c9oclaxv1l8ud%2Fomg-demo.json

CQ3: How can version control of linked geometry descriptions be realised?

Especially during the collaboration process, the version control of geometry – and any content in general – is of utmost importance. Thus, OMG level 3 addresses this topic and can be used to store information about different versions in two different ways, depending on the type of the geometry description: simple or complex (see Fig.3 and 4).

Since the question when a new geometry state should be created and when old ones should be changed is not addressed fully in this paper, this step aims to give better insights into the proposed methodology. It thus also demonstrates how OMG can be combined with OPM.

CQ4: Which are geometries of the same geometry context?

By introducing the `omg:GeometryContext` class, geometries and geometry descriptions can be put in context to ease the extraction of an entire geometric building model for specific use cases (see Fig. 2 and 3). This step demonstrates how such contexts can be added, manipulated and deleted.

CQ5: How can geometries be complemented or transformed?

The properties `omg:transformsGeometry` and `omg:complementsGeometry` allow the definition of complemented and transformed geometries (see Fig. 2). With this definition, it is easy to identify such geometries for further processing during data extraction, exchange or processing.

CQ6: What are the dependencies between properties and geometry?

With the introduced `omg:isExplicitlyDerivedFrom` and `omg:isImplicitlyDerivedFrom` properties and their concepts, meaningful connections between geometry and non-geometric properties can be created. For explicit derivations, the relation between the derived property and the origin's property states can also be inferred by the `omg:hasInferredPropertyState` property, facilitating parametric descriptions (see Fig.6 and 7). The work-flow for doing this is demonstrated in this step of the demo.

CQ7: What is the current timestamp of a geometry state?

This question can be answered in two different ways for simple, non-RDF-based and complex, RDF-based geometry descriptions. Simple geometry descriptions will require a new geometry state for each submitted change which leads to a simple SPARQL query to extract the timestamp. For complex geometry descriptions, however, this becomes more complicated: A new geometry state is only introduced, when the description changed in its used objects, while changes in geometric properties should be handled using the OPM. Thus, all property states of a geometry state must be evaluated – as well as the geometry state itself – regarding their timestamps and the most current timestamp needs to be returned. This process is simplified by the `omg:containsPropertyState` property.

CQ8: Where are discrepancies within a graph?

Finally, it must be possible to identify data discrepancies such as outdated geometry states that are still labelled as `omg:CurrentGeometryState`. Since this is the most complex and needed functionality of the OMG, tab 8 of the *SPARQL-visualizer* demo shows step-by-step how this topic can be handled with the introduced concepts.

In this demo, it is shown how one column element can be connected to multiple, derived (simple) geometries under version control. Beginning with one STEP geometry, a derived OBJ geometry, followed by a derived COLLADA geometry are added. On geometry node level, possibilities for further derivations are modelled. Since the OBJ geometry is for visualising purposes only in this scenario, it can only be derived. Its possible origins are both, the STEP and COLLADA geometry. The STEP and COLLADA geometries can be derived from each other, as both may be used in modelling software applications. The resulting graph on geometry node level can be seen in Fig. 8.

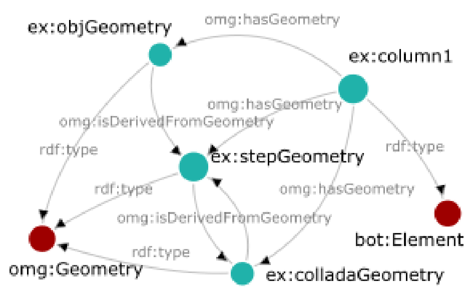

Figure 8: Example for geometry dependencies.

At this time, each geometry also has one geometry state – classified as `omg:CurrentGeometryState` as well as `omg:GeometryState`. Each state contains information about their time of generation, their simple geometry description and their origin of derivation. An overview of the geometry states without discrepancies can be found in Fig. 9. For better visualisation, the connection towards their geometry nodes are not shown in this picture.

This far, no data discrepancies exist. However, in the next step, a new geometry state for the COLLADA geometry is inserted – this may happen when manual changes are made to the description, e.g. during the planning and modelling phases of a building. Since this state contains changes that

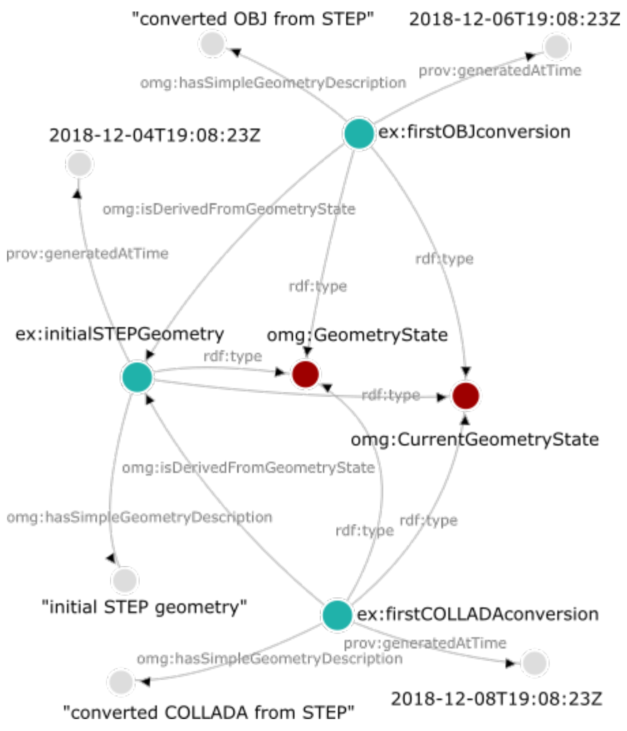

Figure 9: Geometry states without discrepancies.

are not part of the STEP and OBJ geometry states, data discrepancies occur. In order to locate such problems, the data must be queried for them, as is shown in Lst. 2.

```
@prefix omg: <https://w3id.org/omg#> .
@prefix prov: <http://www.w3.org/ns/prov#> .

SELECT DISTINCT ?outdatedGeometry ?originGeometry
WHERE{
#Find all omg:CurrentGeometryState that are either derived from
   or deriving geometry states that are not a
   omg:CurrentGeometryState (and thus outdated)
  ?outdatedGeometryState a omg:CurrentGeometryState ;
    ^omg:isDerivedFromGeometryState|omg:
      isDerivedFromGeometryState ?anyState .
  FILTER NOT EXISTS {
    ?anyState a omg:CurrentGeometryState .
  }
#Any geometry that either contains the previously identified
   outdated omg:CurrentGeometryState or has a geometry state
   that has been derived from it are also outdated.
  ?outdatedGeometry omg:hasGeometryState|omg:hasGeometryState/
     omg:isDerivedFromGeometryState ?outdatedGeometryState ;
    omg:isDerivedFromGeometry ?originGeometry .
#To ensure that only those origins of change and outdated
   geometries are listed that are suitable as an origin of
   change (that are not outdated) respectively are truly
   outdated, the timestamps are compared
  ?outdatedGeometry omg:hasGeometryState ?someOutdatedState .
  ?someOutdatedState a omg:CurrentGeometryState ;
    prov:generatedAtTime ?outTime .
  ?originGeometry omg:hasGeometryState ?originState .
  ?originState a omg:CurrentGeometryState ;
    prov:generatedAtTime ?originTime .
  FILTER (?originTime > ?outTime)
}
```

Listing 2: SPARQL query to find data discrepancies.

In the given example of the demo, this query will return the STEP and the OBJ geometry as outdated and the COLLADA geometry as possible origin for each. After the STEP geometry is updated and the same query is run again, the OBJ geometry will return two possible origins for new derivations: The COLLADA and the STEP geometry. Once all geometries are updated, this query will not yield any results. In order to have these queries working, it is essential that the `omg:isDerivedFromGeometryState` property is used properly.

Conclusions

Within the introduced OMG ontology, a unified approach for linking geometry to building objects, including the definition of dependencies between geometry and non-geometric properties, is proposed. Furthermore, the OMG allows users to decide individually in which detail this connection should be made. For more details, it is possible to define dependencies between multiple geometry representations as well as describing the changes that were made to the description. The OMG is an abstract, upper level ontology that can be extended for specific needs and requirements, while the core of the ontology should not be changed after a mutual agreement within the community on its architecture was made. This will help to ease the integration of geometric descriptions into a Semantic Web context and therefore contribute to establishing Linked Data for the built environment.

However, the schema and its concepts are rather complex, making some sort of middleware necessary. Such a middle-ware should help users to apply the concepts of OMG without having to fully understand how the methodology works. This affects the creation and comparison of geometry and geometry state nodes to identify and prevent data discrepancies, as well as the interpretation of the concepts for geometry transformation and complementation. The latter is defined in the schema, but for interpretation or extraction of the geometry, this definition must be processed to create one singular geometry description that can be used directly in software applications. In this context, it should also be mentioned that, even though it is part of the results of our requirement analysis, the OMG currently does not provide means to mark or identify geometry descriptions as read-only for cases, where potential changes cannot be transferred back to the data pool.

Furthermore, the alignment of the OMG, especially to BOT and OPM, is currently a proposal and must be discussed, e.g. within the W3C LBD CG, and finalised. Another challenge are the rather generic properties to attach the actual geometry description. Since these properties contain no information about the used geometry format and other metadata as the up-axis, the integration of the description into software applications is hindered. Because of this issue, the FOG is introduced to create subproperties for specific geometry formats, including their versions.

In future research, the aforementioned open issues should be addressed and, most importantly, the OMG should be used in practical use cases to evaluate its applicability and validate its concepts. To further enhance the integration of geometry into the Semantic Web, parametric descriptions must be investigated in more detail. At this time, the OMG only defines whether dependencies exist, but do not give any means to describe how these dependencies can be resolved. Thus, an approach to define the exact parametric coherences should be created.

In the aspect of the OMG's character as upper level ontology, a study to evaluate its applicability in other fields, as robotics, mechanical engineering, etc., needs to be conducted. Based on the results of such a study, further alignments of the OMG towards other, upper level ontologies should be examined.

Acknowledgements

This work is part of the research project EnOB: SCOPE, founded by the German Federal Ministry for Economic Affairs and Energy (BMWi).

References

Bonduel, M. (2018), 'Towards a PROPS ontology'. Date accessed: 2019-03-20.
URL: *https://github.com/w3c-lbd-cg/lbd/blob/gh-pages/presentations/props/presentation_LBDcall_20180312_final.pdf*

Bonduel, M., Wagner, A., Pauwels, P., Vergauwen, M. & Klein, R. (2019), Including widespread geometry formats in semantic graphs using rdf literals, *in* 'Proceedings of the European Conference on Computing in Construction (EC3 2019)', Chania, Crete, Greece.

Krijnen, T. & Beetz, J. (2017), 'An IFC schema extension and binary serialization format to efficiently integrate point cloud data into building models', *Advanced Engineering Informatics* **33**, 473–490. DOI: 10.1016/j.aei.2017.03.008.

Lefrançois, M. & Zimmermann, A. (2016), Supporting Arbitrary Custom Datatypes in RDF and SPARQL, *in* H. Sack, E. Blomqvist, M. D'Aquin, C. Ghidini, S. Ponzetto & C. Lange, eds, 'The Semantic Web. Latest Advances and New Domains. ESWC 2016. Lecture Notes in Computer Science', Vol. 9678, Springer, Cham, pp. 371–386. DOI: 10.1016/j.autcon.2016.10.003.

Mirtschin, Jon (2018), 'GeometryGym: OpenBIM tools for Architects, Engineers and the Construction Industry'. Date accessed: 2019-03-20.
URL: *https://geometrygym.wordpress.com/*

Ohori, K., Ledoux, H., Biljecki, F. & Stoter, J. (2015), 'Modeling a 3D City Model and Its Levels of Detail as a True 4D Model', *ISPRS International Journal of Geo-Information* **4**(3), 1055–1075. DOI: 10.3390/ijgi4031055.

Pauwels, P., Van Deursen, D., de Roo, J., Van Ackere, T., de Meyer, R., Van de Walle, R. & Van Campenhout, J. (2011), 'Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* **25**(4), 317–332. DOI: 10.1017/S0890060411000199.

Pauwels, P., Zhang, S. & Lee, Y.-C. (2017), 'Semantic web technologies in AEC industry: A literature overview', *Automation in Construction* **73**, 145–165. DOI: 10.1016/j.autcon.2016.10.003.

Pittet, P., Cruz, C. & Nicolle, C. (2014), 'An ontology change management approach for facility management', *Computers in Industry* **65**(9), 1301–1315. DOI: 10.1016/j.compind.2014.07.006.

Rasmussen, M. H., Frausing, C. A., Hviid, C. A. & Karlshøj, J. (2018), Demo : Integrating Building Information Modeling and Sensor Observations using Semantic Web, *in* M. Lefrançois, R. Garcia Castro, A. Gyrard & K. Taylor, eds, 'Proceedings of the 9th International Semantic Sensor Networks Workshop co-located with 17th International Semantic Web Conference (ISWC 2018)', number 1, Monterey, CA, United States, pp. 48–55. CEUR: Vol-2213/paper4.pdf.

Rasmussen, M. H., Lefrançois, M., Bonduel, M., Hviid, C. A. & Karlshøj, J. (2018), OPM: An ontology for describing properties that evolve over time, *in* '6th Linked Data in Architecture and Construction Workshop (LDAC), CEUR Workshop Proceedings', Vol. 2159, London, UK, pp. 23–33. ISSN: 16130073.

Wagner, A., Möller, L. K., Leifgen, C. & Rüppel, U. (2018), SolConPro: Describing multi-functional building products using semantic web technologies, *in* J. Karlshøj & R. J. Scherer, eds, 'EWork and eBusiness in architecture, engineering and construction: proceedings of the 12th European Conference on Product and Process Modelling (ECPPM 2018)', 12, CRC Press, pp. 447–455. ISBN: 978-0-429-01364-5.

Zhang, C., Beetz, J. & de Vries, B. (2017), 'BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data', *Semantic Web Journal* **1**(0), 1–17. DOI: 10.3233/SW-180297.