Including widespread geometry schemas into Linked Data-based BIM applied to built heritage

Mathias Bonduel MSc
PhD student, Department of Civil Engineering, Katholieke Universiteit Leuven, Ghent, Belgium (Orcid:0000-0002-3313-924X)
(corresponding author: mathias.bonduel@kuleuven.be)

Anna Wagner MSc
Research Assistant, Department of Civil and Environmental Engineering, Sciences, Technische Universität Darmstadt, Darmstadt, Germany (Orcid:0000-0001-6941-7144)

Pieter Pauwels PhD
Associate Professor, Department of the Built Environment, Eindhoven University of Technology, Eindhoven, the Netherlands (Orcid:0000-0001-8020-4609)

Maarten Vergauwen PhD
Associate Professor, Department of Civil Engineering, Katholieke Universiteit Leuven, Ghent, Belgium (Orcid:0000-0003-3465-9033)

Ralf Klein PhD
Assistant Professor, Department of Civil Engineering, Katholieke Universiteit Leuven, Ghent, Belgium (Orcid:0000-0003-1023-7629)

A reliable data exchange – often including geometry-related data – between stakeholders is crucial in construction projects. In this regard, data exchange frameworks built on Linked Data principles are very promising for combining disparate data sets. However, existing proposals to combine geometry and Linked Data either demand dedicated applications or support only a limited number of common geometry schemas. If any existing geometry schema could be used in a Linked Data context, error-prone geometry conversions are avoided and stakeholders do not need to invest in new geometry engines. In this paper, the applicability of Resource Description Framework (RDF) literals for including a wide variety of existing geometry schemas is studied and applied in a built heritage context. The uniform linking pattern and related terminology of the Ontology for Managing Geometry are used to implement this approach. Subsequently, the File Ontology for Geometry formats and Geometry Metadata Ontology are developed to ease the reuse of linked geometry descriptions. The effectiveness of the entire data structure is demonstrated in a built heritage case study project. The receiving party is able to create successfully a coordinated view – using a demo web application – on shared, but disparate, RDF data sets containing geometry descriptions.

Notation

\[ M = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix} \]

\[ m_{ij} \] matrix element of \( 4 \times 4 \) matrix \( M \) in row \( i \) and column \( j \)

\[ X_i \] \( x \)-coordinate in coordinate system \( i \)

\[ Y_i \] \( y \)-coordinate in coordinate system \( i \)

\[ Z_i \] \( z \)-coordinate in coordinate system \( i \)

1. Introduction

Similarly as in other construction-related domains, such as building and infrastructure design, researchers in the built heritage sector actively study solutions to improve digital building data management (Myers et al., 2016; Pocobelli et al., 2018). This includes concepts such as three-dimensional (3D) surveying and semantic modelling (digital twin), database thinking and controlled data exchange, all closely related to building information modelling (BIM). According to previous studies, the introduction of BIM in the built heritage sector (heritage/history BIM or HBIM) can improve the data management aspects of projects (Bruno et al., 2018). The overall efficiency of tasks can be increased by supporting a closer collaboration between stakeholders. The management of both geometrical and semantic descriptions of heritage buildings is crucial for HBIM to be successful.

Typical built heritage projects for restoration or conservation can often be considered more complex than regular greenfield construction projects (Simeone et al., 2019). First, there is an existing construction that has to be analysed for historical importance and current state of conservation based on a disparate set of existing documents, monitoring data and site surveys. Second, a wider variety of stakeholders with different background knowledge, workflows and applications can be participating in such a project. Besides the owner, restoration planners and contractors, other relevant stakeholders are the public conservation bodies, heritage inspectors, facility managers and researchers dealing with historical architecture, all included in Figure 1.

1.1 Existing BIM applications and schemas

The potential of conventional BIM applications and the vendor-neutral Industry Foundation Classes (IFC) schema (ISO, 2018) in the built heritage domain has been studied in several articles (Bruno et al., 2018; López et al., 2018; Pocobelli et al., 2018). Case studies demonstrate that these BIM environments can be used for heritage projects to a certain degree only, because of general and specific heritage-related shortcomings.

In general, they lack a standardised query language, in contradiction to many database systems, and their data structure is either proprietary in the case of commercial BIM authoring tools or rather complex in the case of IFC (Quattrini et al., 2017). Collaboration between stakeholders is based on the exchange of entire files in a...
proprietary or neutral data schema. The quality of IFC files heavily depends on the quality of IFC exporters and importers of proprietary BIM applications.

Considering built heritage, both IFC and schemas of proprietary BIM systems miss specific classification methods for heritage building objects – for example, flying buttresses and gargoyles. Additionally, such BIM data cannot easily be connected to data coming from other applications, including data from geographic information systems (GISs), sensors, heritage inventory systems, material databases, damage inspections and historical sources (Pauwels et al., 2017a; Simeone et al., 2019). Complex metadata statements related to the provenance, context, reliability and uncertainty of the data are difficult or impossible to include in the model, while omnipresent in heritage projects (De Luca et al., 2011; Gómez-Romero et al., 2015). To continue, the 3D geometry description of a building object is intrinsically connected to the semantic object in conventional BIM authoring tools. Subsequently, every building object needs exactly one geometry description, before it is actually defined in the BIM database and can be enriched with additional information. At least in a heritage context, a digital building element should be able to have multiple geometry descriptions. They can be relevant for specific purposes, coming from different origins, and/or reflect the object at certain moments throughout its history. Examples are detailed point clouds or complex meshes derived from surveys, lines as part of existing two-dimensional (2D) plans or simplified 3D geometry based on available survey data or plans (Fai et al., 2011). As conventional BIM applications are originally developed for the design of new buildings, they often miss practical modelling commands to create non-regular 3D geometries typical of certain heritage buildings (e.g. a slanting irregular stone wall, cross-vaults and complex ornaments) (Garagnani and Manferdini, 2013; Yang et al., 2019).

1.2 Applying Linked Data to BIM

As a consequence of these limitations of existing BIM applications and schemas, groups of construction and heritage researchers investigate Linked Data and Semantic Web technologies as an alternative approach to BIM (Pauwels et al., 2017a), specifically HBIM (McGlinn et al., 2019; Quattrini et al., 2017; Simeone et al., 2019; Yang et al., 2019). Linked Data is a part of the Semantic Web technology stack based on World Wide Web Consortium (W3C) web standards (Bizer et al., 2009), including Resource Description Framework (RDF), RDF Schema (RDFS), Web Ontology Language (OWL) and Sparql Protocol and RDF Query Language (Sparql). Multiple subject–predicate–object statements or RDF triples form a directed graph. These triples can be stored in RDF files or dedicated databases and queried using Sparql. Each node in an RDF graph is in general identified with a unique Uniform Resource Identifier (URI).

As a result, nodes in one RDF graph can link to nodes included in other RDF data sets, allowing easily combining disparate RDF triples coming from different actors.

The RDF data model is in essence domain-agnostic, but domain-specific terminology (classes, properties and datatypes) can be defined in a web ontology using RDFS or Owl. Such ontologies can include human-readable labels and definitions for defined terminology, as well as relations to related terminology and formal logics (Guarino and Oberle, 2009). Existing public ontologies for buildings are either considerably large and complex (e.g. ifcOwl (Pauwels and Terkaj, 2016), the web ontology representation of IFC) or more concise and modular. The second group of ontologies is considered easier to reuse, extend and implement in applications. Based on IFC, the ifcOwl ontology combines geometrical and non-geometrical aspects in one data schema. When converting an IFC building model into an ifcOwl graph, the geometry of each individual building object is defined as a large set of RDF triples (Pauwels et al., 2017b). The benchmark duplex house IFC model, for example, results in an ifcOwl RDF graph of 227 143 asserted triples, as measured by Bonduel et al. (2018a). To the best of the authors’ knowledge, there exists no application that can deal directly with the geometrical part of ifcOwl. Alternatively, examples of more modular building ontologies are the Building Topology Ontology or BOT (Rasmussen et al., 2017) and other related ontologies for property management and product classification under development within the Linked Building Data Community Group of the W3C (Bonduel et al., 2018a). More recently, the compatible Damage Topology Ontology (DOT) module and several extensions were developed for modelling the concept of construction damage (Hamdan et al., 2019).
supported geometry schemas different from WKT and GML. In contrast, heritage stakeholders currently use a wide variety of existing geometry schemas – both open or proprietary, 2D or 3D, binary or text-encoded – for example, OBJ, PLY, Step, STL, glTF, DWG and ES7. These geometry schemas are supported in a large amount of available computer-aided design (Cad) and/or point cloud applications (Pauwels et al., 2011).

1.3 Paper outline
This paper contributes to the representation and linking of built heritage geometry in a Linked Data BIM environment. The research focuses on the need for a structured data exchange between built heritage stakeholders that is crucial for the collaboration during both restoration and maintenance processes. More specifically, a method for including widespread geometry schemas in RDF graphs using RDF literals is studied. Additionally, a flexible and domain-independent linking method is applied to link any type of object to their corresponding geometry description. The envisaged data structure has to be clear and explicit and provide enough context to make the exchange and reuse of geometry possible in a collaborative built heritage environment. Finally, a workflow for creating and reusing geometry descriptions in RDF graphs of built heritage projects is discussed.

The related work in this particular research domain on the crossroads of Linked Data, built heritage and geometry is included in Section 2. Section 3 discusses a method for reusing existing geometry schemas in RDF graphs based on RDF literals and a universally applicable linking method for connecting semantic building objects to their geometry descriptions. Section 4 contains the results of a case study built heritage project where building data and linked geometry descriptions are exchanged based on the data structure discussed in Section 3. Finally, the results of the case study are discussed in Section 5 before concluding the research findings in Section 6.

Throughout the paper, multiple web ontologies are mentioned by their prefix or short name. The namespaces and prefixes corresponding with these ontologies are listed in Table 1. The first group of ontologies is mentioned in related work and (partially) reused in the presented data structures. The second group refers to the main terminology modules applied in this research, while the last group contains the namespace and prefix for the instances in the example data sets.

### 2. Related work

This section discusses research related to built heritage, geometry and Linked Data. Two groups of relevant articles are distinguished during the literature review, each with their own perspective on the research domain. The first group starts with applying the HBIM concept in conventional BIM authoring tools and moves towards a Linked Data-based HBIM to overcome limitations in existing tools and data formats. On the other hand, the primary goal of research in the second group is to document heritage buildings in a built heritage inventory system, for which Linked Data technology is used directly. It is relevant to study both perspectives, as they try to achieve a similar goal – namely, creating a meaningful overview of otherwise unconnected data sources related to the heritage building. Furthermore, methods developed by the second group are not bounded by the strict database structure in conventional BIM applications, as discussed in the previous section.

#### 2.1 HBIM perspective

In the paper by Quattrini et al. (2017), a workflow is described to create a HBIM database in the Autodesk Revit software program based on point cloud data, enriched with custom Revit properties that have a counterpart in the terminology of a separately developed web ontology. At the end of the modelling process, the database is exported in the neutral IFC format, which is subsequently transformed into an RDF graph following the ifcOwl ontology using the IFC-to-RDF converter (Pauwels and Oraskari, 2016). As the ifcOwl structure is considered too complex, a simplification phase is introduced using their new but undocumented domain ontology. The RDF graph is then loaded in the developed BIMExplorer web application to visualise dynamically the 3D model based on faceted browsing while communicating with the RDF graph. It is unclear how the visualised geometry is included or referenced in the RDF graph, as there is no example RDF data set and the developed ontology is not shared online.

<table>
<thead>
<tr>
<th>Table 1. Namespaces of the referenced ontologies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prefix</strong></td>
</tr>
<tr>
<td>rdf</td>
</tr>
<tr>
<td>rdfs</td>
</tr>
<tr>
<td>owl</td>
</tr>
<tr>
<td>xsd</td>
</tr>
<tr>
<td>fcowl</td>
</tr>
<tr>
<td>geom</td>
</tr>
<tr>
<td>brep</td>
</tr>
<tr>
<td>geos</td>
</tr>
<tr>
<td>strdf</td>
</tr>
<tr>
<td>ecml</td>
</tr>
<tr>
<td>chml</td>
</tr>
<tr>
<td>bot</td>
</tr>
<tr>
<td>beo</td>
</tr>
<tr>
<td>dot</td>
</tr>
<tr>
<td>qudt</td>
</tr>
<tr>
<td>unit</td>
</tr>
<tr>
<td>epsg</td>
</tr>
<tr>
<td>prox</td>
</tr>
<tr>
<td>dct</td>
</tr>
<tr>
<td>3dmo</td>
</tr>
<tr>
<td>omg</td>
</tr>
<tr>
<td>fog</td>
</tr>
<tr>
<td>gom</td>
</tr>
<tr>
<td>inst</td>
</tr>
</tbody>
</table>
The workflow presented in the paper by Simeone et al. (2019) starts from an Autodesk Revit heritage model, which is connected to an RDF knowledge base in the ontology-editing application Protégé using the Revit DB Link plug-in. The BIM Semantic Bridge application was developed to map concepts and manage the bidirectional exchange of data between the Revit database and the RDF graph stored in Protégé. The geometry of building spaces and elements resides entirely in the Revit model and is not represented in the RDF database.

A mixed approach is followed in the work of Yang et al. (2019): a 3D heritage model is created in Revit and exported in IFC, which is converted into RDF following ifcOwl using the IFC-to-RDF converter. Afterwards, the RDF graph is enriched in Protégé with non-geometric data. Links between the Revit and RDF database are established using a custom script running in the Autodesk Dynamo visual programming environment. The geometry is manually created in Revit based on point cloud data (‘manual scan-to-HBIM’) or in Rhinoceros Cad based on meshes fitted on point clouds (‘semi-automated mesh-to-BIM’). In the second case, the static geometry is transferred from Rhinoceros to Revit.

In the EU project Inception, Linked Data principles are applied when creating data sets related to built heritage and include geometry descriptions (McGlinn et al., 2019). First, a digital heritage model is created in a BIM authoring tool and exported to IFC. Subsequently, the IFC model is converted into two Linked Data graphs: one for non-geometrical aspects according to the project H-BIM ontology and one for geometry descriptions according to the Geom ontology. Instances in the two graphs are connected using a third ontology. Utilising the Geom geometry schema, the geometry description of each building object is stored as a set of nodes and relations between these nodes forming an RDF graph. Besides IFC geometry, the Inception framework can convert other geometry schemas such as Collada and DWG into an RDF graph according to Geom. While the GEOM ontology is publicly available, the H-BIM and unnamed ‘connection’ ontology are not shared online to the authors’ knowledge. Specific applications are necessary to visualise and edit this type of geometry descriptions accordingly. The authors report in the related section of the paper by McGlinn et al. (2019) that using Geom or related ontologies (e.g. OntoBrep) as a geometry schema might result in potentially longer loading times compared with discrete geometry descriptions in widespread Cad formats.

2.2 Heritage inventory perspective

Kuroczynski et al. (2016) describe a web application based on Linked Data principles for lost heritage buildings and the digital reconstruction aspect specifically. Their RDF graph is constructed using terminology from the proposed Cultural Heritage Markup Language web ontology, which extends the Erlangen Cidoc Conceptual Reference Model (CRM) Reference Model (CRM), the web ontology mirror of the Cidoc CRM schema (Götz et al., 2008) for documenting heritage objects as standardized by ISO International Organization for Standardization (ISO). The geometry of the reconstructed building is created in Cad applications and is based on different sources such as point cloud surveys of ruins, old plan documents and photographs. The reconstruction activity nodes are connected to RDF literals containing the references to the source files and the reconstructed geometry descriptions in a common Cad format.

The work of Messaoudi et al. (2017) focused on the management of deterioration of existing heritage buildings based on point clouds. The internal data structure of the presented data-management application follows their (unpublished) Cultural Heritage web ontology. It is unclear how the 3D point cloud geometry coming from surveys is stored in the dedicated database. References to annotations on this point cloud geometry, including 3D indexes, are included in a MySQL relational database together with a URI of the corresponding annotation node in the RDF database of the system. A virtual RDF database is defined on top of the MySQL database, allowing the querying of the MySQL database as an RDF graph.

2.3 Conclusion

All studied articles mention geometry and its relevance in the built heritage sector, but the relation between geometry descriptions and Linked Data graph(s) is not widely discussed. It is difficult to study proposed data structures in detail, as most researchers do not publish their developed web ontologies and sample data online.

In research from the first group, it is proposed to start with conventional BIM systems for the geometrical modelling, while Linked Data enrichment is possible only in later phases. As a result, such processes remain dependent on the usage of conventional BIM tools or neutral formats such as IFC throughout the entire workflow. Only the GEOM section in the paper by McGlinn et al. (2019) mentions a publicly available data structure for geometry in a Linked Data context, but their rather complex solution based on RDF graphs describing the geometry needs the development of dedicated applications for creating and (re)using these geometry descriptions. Other, more agile workflows and data structures are studied in this paper to make the exchange of building data easier – for example, by reusing widespread geometry schemas and existing Cad applications.

The second group provides interesting insights by linking resources in RDF graphs to external geometry files created with regular Cad applications or based on laser scan or structure-from-motion (SIM) surveys. The discussed applications and data structures are, however, primarily focused on built heritage documentation and management. In other words, they are not designed to be used in a collaborative restoration and maintenance process – that is, they miss modular, commonly agreed data structures and methods for controlled data exchange based on RDF. As an example, the linking method described in the paper by Kuroczynski et al. (2016) does not refer to the used geometry schema of the linked geometry description. The RDF graph...
misses additional contextual metadata, such as the used geometry schema and version, applied coordinate systems (CSs) and derived and grouped geometry descriptions for other stakeholders to reuse conveniently the geometry in their applications. One possible reason for the limited discussion of geometry is the fact that geometry, in certain cases, can be considered relevant for other vertical domains than construction engineering – for example, geosciences, mechanical engineering, biosciences and e-commerce. The data structures for describing and connecting geometry in a Linked Data context as such are not often the main focus of researchers working in the built heritage domain. When a new method for describing and connecting geometry in a Linked Data context is developed, it is worthwhile studying if it can be applied in multiple vertical domains. Similarly, a domain-independent and shared web ontology for linking geometry description to objects can be valuable. Such an approach (a) avoids the establishment of different islands of standardisation per vertical domain and (b) stimulates the creation of concise and modular ontologies without overlapping terminology defined for individual vertical domains.

3. Geometry and RDF literals

To include geometry descriptions in widespread geometry schemas in Linked Data graphs, the definition of graph data structures and related terminology is needed. This section starts with an introduction of RDF literals for embedding or referencing such geometry descriptions. In Section 3.2, a universal linking method and related terminology is described. Finally, Section 3.3 discusses how CSs, units and other metadata can be linked to geometry descriptions. The first two sections summarise the findings of an earlier work (Bonduel et al., 2019) and includes aspects of the work presented by Wagner et al. (2019). The interested reader can find more details and argumentation for the proposed data structure in the respective papers.

The ontologies described in this section are retrievable in RDF and Hypertext Markup Language by way of their namespace and include references to example data sets and queries to demonstrate the envisaged graph patterns.

3.1 The potential of RDF literals

First, the usage of RDF literals to embed geometry descriptions as text snippets is studied as an alternative to complex geometry descriptions expressed as RDF graphs with a separate dedicated toolchain (Bonduel et al., 2019). Based on the study of the relevant W3C Recommendations, it is concluded that RDF literals can be used to store the content of any existing geometry schema as long as the content is valid Unicode text. In the case of binary geometry schemas, the original geometry file has to be text-encoded – for example, using the hexadecimal and Base64 encoding scheme. Text-based geometry schemas can be wrapped in an RDF literal, but attention has to be given to correctly escaping line breaks and quotation marks.

An alternative approach to including geometry descriptions of widespread schemas is to reference external geometry files by way of their file location stored in an RDF literal. This method is also described in the Information Container for Data Drop ISO standard, which is currently under review (ISO, 2019), and in the publication by Kuroczyński et al. (2016), as mentioned in Section 2. It is not advisable to use the file location as a graph node, as it would imply the identifier of the geometry description and the file location to be considered the same. If the geometry file would change location, the URI of the node would be updated, which is considered a bad practice, as URIs are expected to be persistent (Sauermann et al., 2008).

Using RDF literals as described in previous works (Bonduel et al., 2019; Wagner et al., 2019) allows the built heritage sector to reuse the myriad of existing widespread geometry schemas and related toolsets. In other words, conversions of common geometry schemas into complex RDF-based geometry are avoided and the implementation efforts in existing applications are minimal.

3.2 A unified method for linking geometry

In the second step, the need to define practical Linked Data patterns to link graph nodes – for example, building objects – to their geometry descriptions is considered. A list of practical requirements related to the reuse of geometry descriptions in a collaborative environment is discussed by Bonduel et al. (2019). These requirements include the need to be explicit about the used geometry schema and – in the case of binary geometry schemas – the used encoding scheme.

Additionally, a graph node representing a certain object can link to multiple geometry descriptions (e.g. evolution of geometry over time, different origins or types of the geometry and different geometry schemas). It should also be possible to add relevant metadata in RDF to make the reuse of geometry descriptions by other actors convenient. Ideally, the envisaged data structure should be applicable to link objects to geometry descriptions independently of the used method for storing the geometry description in a Linked Data context.

Based on the requirements listed in the previous paragraph and inspired by existing implementations, a data structure is described that stimulates reuse and even automatic processing of geometry. The datatype of the RDF literal is used as a first differentiator (see Figure 2). The xsd:string datatype is attached if the RDF literal contains an embedded geometry description of a text-based geometry format. When the embedded geometry description is binary, the selected text-encoding is mentioned using the fixed datatypes xsd:base64Binary or xsd:hexBinary or custom datatypes for other encoding schemes. In the case of referenced external geometry files, the datatype xsd:anyURI is utilised.

Secondly, the uniform geometry linking mechanism from the Ontology for Managing Geometry (OMG) module (Wagner et al., 2019) can be used; the datatype property omg:hasSimpleGeometryDescription allows linking a node to a geometry description embedded in or referenced from an RDF literal, while the object property omg:hasComplexGeometryDescription can be applied to link to geometry descriptions expressed as an RDF graph. As these OMG properties are not specific concerning
the used geometry schema in the geometry description, they are extended in the File Ontology for Geometry formats (FOG) module. Each geometry schema is represented as a subproperty of omg:hasSimpleGeometryDescription or omg:hasComplexGeometryDescription, as demonstrated in Figure 3. These two properties are made more specific related to the version of the geometry schema and – if applicable – one of the multiple associated files or file extension. Although the FOG module is already quite extensive, it is impossible to include every available geometry schema upfront. The development of this ontology is therefore considered a community effort, as users are encouraged to suggest geometry schemas by way of the public GitHub repository of Bonduel et al. (2020a).

With the datatype and the used FOG properties, the geometry can be processed automatically by the receiving application, as the potential encoding is known and the precise geometry schema and version is specified. With this information, the right decoder and geometry loader can be called. An extra benefit of explicitly defining the used geometry schema is that client applications can now ask specifically for geometry that can be handled, avoiding the exchange of redundant data.

Similarly as suggested by Rasmussen et al. (2018) for the modelling of properties, OMG enables three levels of complexity for linking objects to their geometry description (Figure 4). As such, users can trade modelling complexity for extra functionality by selecting a certain level. OMG level 1 directly connects the object to its geometry description, while OMG level 2 introduces an intermediate omg:Geometry instance node between the object and the actual geometry description. The geometry node can be used to add metadata related to the geometry description. The third level introduces an omg:GeometryState instance node between the geometry node and the geometry description to enable versioning of geometry in the graph. This extra node can be used to differentiate between versions of the geometry object (e.g. design iterations or to indicate real geometrical changes over time). The existing property prov:generatedAtTime is used to indicate the creation time of the omg:GeometryState instance, while its omg:CurrentGeometryState subclass is used to quickly retrieve the most recent version. More details on geometry version management can be found in the paper by Wagner et al. (2019).

OMG also contains a grouping mechanism for geometry descriptions by linking geometry nodes to instances of omg:GeometryContext (e.g. grouping in as-designed and as-is geometry, grouped per originating stakeholder, per analysis etc.) and provenance-related terminology to define relations between geometry nodes – for example, omg:isDerivedFromGeometry. This last property is particularly relevant in the case of built heritage, where volumetric 3D geometry is often derived from point clouds or existing 2D plans. Additionally, the original point clouds and meshes often have to be simplified before they can be used in some geometry applications, which is another derivation.

As the proposed linking method concerns geometry aspects only, it can potentially be applied in any other vertical domain that requires geometry descriptions to be included in a Linked Data context. In more detail, the properties for linking geometry defined in the OMG/FOG ontology modules have no rdfs:domain restrictions defined, meaning that they can be used to link any object to a related geometry description. A broader uptake of this linking method and accompanying terminology by developers employed in different domains might be beneficial regarding the costs and quality of supporting application plug-ins.

### 3.3 CSs, length units and general metadata

Besides OMG and FOG, a third ontology module named Geometry Metadata Ontology (GOM) is developed to deal with CSs, length units and general metadata, relevant for geometry. The module is designed to be at least compatible with OMG/FOG and their related graph patterns.

Every geometry description is defined in exactly one CS. When multiple stakeholders participate in a project, they might each define their geometry descriptions in different CSs. Considering building projects, custom 3D Cartesian CSs are often used in Cad software, while GIS applications also make use of, for example, ellipsoidal...
If a unified view of disparate geometry descriptions is created, it is necessary to know the CS for each geometry description. In addition, the different custom CSs of the relevant geometries need to be either directly registered to each other or georeferenced by defining a registration towards a predefined coordinate reference system (CRS) – for example, Belgian Lambert 2008 (\(X\Y\)) with Ostend height \(Z\). Existing standards and services provide the necessary transformations between different predefined CRSs (OGC, 2001).

If a geometry description is defined in a collaborative Linked Data context, it preferably gets a named CS assigned to it representing either a custom CS or an existing CRS. In the proposed GOM module, the object property gom:hasCoordinateSystem is defined to link geometry to a gom:CoordinateSystem instance (see Figure 5). The URIs for CRSs applied in GeoSparql can be reused – for example, epsg:6190 corresponds to the Belgian Lambert 72 CRS with Ostend height.

A custom Cartesian CS can be registered to another custom Cartesian CS or Cartesian CRS of the same spatial dimensions with an affine transformation, which in most cases is a similarity transformation (translations, rotations and scalings). A 3D transformation can be defined in a \(4 \times 4\) transformation matrix \(M\), the result of a concatenation of translations \(T\), rotations \(R\) and scalings \(S\). In Figure 5, the gom:AffineCoordinateSystemTransformation instance node is connected with gom:toCartesianCoordinateSystem and the destination CS gom:<coordinateSystem-B> using gom:hasTransformationMatrix property. Alternative methods for expressing a transformation could be added. The transformation of coordinates is calculated using Equation 1. A square matrix can be defined in a row-major order or column-major order array, which is added using the corresponding datatypes from GOM. All unregistered geometry is assumed defined in a single unnamed custom CS. In the specific case where a geometry description is represented in a 2D space (e.g. in the Scalable Vector Graphics (SVG) schema, only coordinates \(X\) and \(Y\) are defined) and has to be analysed in a 3D space together with other geometries, the receiving application has to assume that the missing coordinate is equal to zero. As a result, the 2D geometry description can also be defined in a named or unnamed 3D CS.

\[
\begin{bmatrix}
X_b \\
Y_b \\
Z_b \\
1
\end{bmatrix}
= 
\begin{bmatrix}
X_a \\
Y_a \\
Z_a \\
1
\end{bmatrix}
\]
In a general collaboration setting, the plethora of applications to create and use geometry descriptions can have different settings regarding the length unit. This would not be a problem if all used geometry schemas are allowed to include the used units or have a predefined unit. However, several widespread geometry schemas in use do not have defined units. Examples of such schemas are OBJ, PLY and PCD. Each geometry description is already defined in exactly one CS. CRSs already have a certain length unit, defined in their public documentation, while custom CSs need a length unit. In the authors’ approach described in Figure 6, a qudt:Unit instance (QUDT, 2019) can be linked to the gom:CoordinateSystem instance using the gom:hasLengthUnit property. In the case of unregistered geometry, the single unnamed custom CS of this geometry description is assumed to be in SI (metres). The length unit metre is also assumed if a custom Cartesian CS has no connected unit. If a geometry description has a defined length unit from the geometry schema that is different from the unit connected to the related gom:CoordinateSystem, the application reusing this geometry has to scale the geometry description accordingly before doing the transformations between CSs.

Instances of gom:Geometry and gom:GeometryState can be enriched with relevant metadata. Geometry descriptions can be filtered during querying based on such metadata. Properties of popular provenance ontologies such as PROV Ontology (prov:generatedAtTime) and the Dublin Core Metadata Terms (dct:creator) can be reused for generic metadata. Inspired by the 3D Modelling Ontology (Sikos, 2017), more specific metadata properties for geometry are added to GOM. This includes the datatype properties gom:hasVertices (point clouds or meshes), gom:hasEdges (meshes) and gom:hasFaces (meshes) for quantitative measurement values. The object property gom:createdIn links a geometry to a predefined instance of gom:GeometryModellingApplication. Other GOM metadata properties are gom:hasFileSize, gom:hasVolume and gom:hasSurfaceArea. A series of non-disjoint subclasses of gom:Geometry is available for classifying geometry nodes more precisely. Examples are gom:MeshGeometry, gom:PointCloudGeometry and gom:PlanGeometry. Survey geometry of built heritage is often converted to 2D plans and simplified 3D models. To express the represented accuracy of the simplified 3D model based on the Level of Accuracy specification (USIBD, 2016), the methods described by Bonduel et al. (2017) can be used. To express the results of such an analysis, an instance of gom:RepresentedAccuracyAnalysis is linked to the original geometry with gom:hasSurveyGeometry and to the simplified geometry with gom:hasSimplifiedGeometry. The datatype properties gom:
totalRelevant gom:totalOccluded, gom:totalLOA50, ..., gom:totalLOA10 are used to link to percentage values summarising the quantitative results of the analysis. It is also possible to use the gom:total95PercentUpperLimit property to link to an RDF literal containing the deviation limit in mm where 95% of the surface area complies with. The datatype property gom:hasAnalysisFile links to an RDF literal embedding or referencing the file with the detailed analysis results. Relevant settings of the analysis are added using the datatype properties gom:maxOcclusionDistance (in millimetres) and gom:usedLocalModelMethod.

4. Application of Linked Data and geometry for built heritage

In the following section, the technical results from Section 3 are applied on a case study project considering the restoration of the Saint Nicolas Church in Ghent (Belgium). The Gothic church with origins in the thirteenth century is one of the landmarks of the city and is a protected monument. Throughout the long history of the monument, several reparations, changes and restorations have taken place. The City and the Diocese of Ghent are currently planning the next restoration phase, involving both the interior and exterior of the façade of the southern nave-aisles. As is typical during a restoration campaign of a precious heritage building, detailed architectural and historical research is needed to create an overview on the current state of the building and its historical value.

The demo RDF data sets concerning a fictive but realistic case study project are described in Section 4.1 and can be found online on the GitHub repository of Bonduel et al. (2020b). The OMG/FOG linking method is applied to link to RDF literals, and the data sets are enriched with triples expressing CSs, length units and other relevant metadata using terminology from the GOM module. Second, a demo web application is created to demonstrate how disparate RDF graphs with linked geometry descriptions can be queried and used. A live version of the application is available the web app is not the GitHub repo (Bonduel, 2020a), while the source code can be retrieved from the GitHub repository of Bonduel (2020b). Section 4.2 contains the implementation details of this application.

4.1 Case study data sets

The input of 20 stakeholders related to heritage buildings in Ghent was collected during a stakeholder analysis, involving a study of the current workflows and data exchange needs. Among these stakeholders were architects, engineers, contractors, heritage inspectors and members of the facility-management department of the City. The existing workflows do not yet involve Linked Data technology, but its potential was recognised by the stakeholders. Based on these results, an example theoretic workflow is elaborated involving the exchange of geometry in a Linked Data context during a built heritage analysis as part of a larger restoration or maintenance process. Figure 7 contains a schematic overview of the example workflow as a process map. The three

Figure 7. Theoretic process map for a built heritage analysis: exchange of geometry in a Linked Data context. TPF, Triple Pattern Fragments
stakeholders directly involved in this fictive workflow are the architect representing the owner, the regional monument inspection organisation and a local historic research contractor.

In the example workflow, the monument inspector executes a survey of the general architectural state of the building. First, the monument’s building topology is modelled in RDF using the BOT terminology, while the Building Element Ontology is applied to classify building elements. The building instance node is connected to public, sparse point cloud geometries of the exterior offered by the Flemish government (Informatie Vlaanderen, 2014). The results of the monument inspection – that is, a list of the damaged elements and the type of damage – are added to the RDF graph using terminology from DOT and linked to the RDF building description. The geometry of the building elements establishing the church nave and nave-aisles is modelled in a 3D Cad application based on available plan documents followed by the geometry of the listed damages. These geometry descriptions are linked to the corresponding building or inspection-related instances in the RDF graph. In parallel, the historical research contractor undertakes a detailed 3D survey of the interior side, focusing on the described façade and surrounding walls, to study their historical value. The results of the laser scanning and SIM survey are detailed, textured meshes that are linked to the shared RDF building description of the inspector. Relevant areas of these meshes are marked using bounding volumes and linked to the corresponding building element in the RDF graph.

Both stakeholders have their own specific goals and tasks as demanded by the restoration architect. Consequently, they use different geometry applications and geometry schemas. During the linking of the geometry to the RDF graph using OMG/FOG terminology, additional metadata based on the OMG and GOM terminology is provided in the RDF graph to ensure that the geometry descriptions can be reused efficiently in other applications. Based on the valuable input data from these two stakeholders, the architect can start with the description of the restoration tasks, which will later be executed by a selected contractor. The relevance of the developed ontology patterns will be highlighted in the next paragraphs based on tangible examples derived from the case study. Table 2 presents an overview of the different geometry descriptions and their properties used in the case study data sets. How these six example parts with geometry-related triples fit in the overall building topology structure described in RDF using BOT and DOT is visualised in Figure 8. The number of the parts correspond to individual parts of the created data sets discussed in the following paragraphs: parts 1–4 (Figure 9) are related to the monument inspector, while parts 5 and 6 (Figure 10) consider the geometry created by the historic research contractor.

Part 1 (Figure 9(a)) describes how point cloud geometries are connected to the building instance node representing the Saint Nicolas Church. A set of three lidar point clouds in LAZ format are found on the web portal of the Flemish government. After isolating the combined points related to the church from its surroundings, the result is saved in the PCD format. The point cloud geometry descriptions are included in the RDF graph by using RDF literals containing references to the external file locations. By using the omg:isDerivedFromGeometry property, the single PCD is linked to the three original lidar point clouds. All point clouds are defined in the Belgian Lambert 72 with Ostend height (EPSG 6190) Cartesian CRS, which is linked to the geometry nodes. A public predefined URI is reused for this CRS – that is, epsg:6190. No length units have to be added to the graph, as the CRS documentation already defined them to be in metres.

Existing 2D plan documents in DWG format are retrieved by the inspector and converted into an SVG vector image on a scale of 1:500 for reuse by the architect. In part 2 (Figure 9(b)), a DWG and

Table 2. Features of the varied example geometry descriptions of the case study data sets, grouped per example part

<table>
<thead>
<tr>
<th>Part</th>
<th>Description of example geometry</th>
<th>Schema and version</th>
<th>2D or 3D</th>
<th>Binary or text (encoding)</th>
<th>Open or proprietary</th>
<th>Additional parts</th>
<th>Embedded or referenced</th>
<th>CS or CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sparse point clouds, church exterior</td>
<td>LAZ v1.4</td>
<td>3D</td>
<td>Binary</td>
<td>Open</td>
<td>—</td>
<td>Referenced</td>
<td>EPSG 6190</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCD v0.7</td>
<td>3D</td>
<td>Text</td>
<td>Open</td>
<td>—</td>
<td>Referenced</td>
<td>EPSG 6190</td>
</tr>
<tr>
<td>2</td>
<td>Original 2D plans, entire church</td>
<td>DWG v2018</td>
<td>2D</td>
<td>Binary</td>
<td>Proprietary</td>
<td>—</td>
<td>Referenced</td>
<td>Custom CS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVG v1.1</td>
<td>2D</td>
<td>Text</td>
<td>Open</td>
<td>—</td>
<td>Referenced</td>
<td>Custom CS2</td>
</tr>
<tr>
<td>3</td>
<td>Volumetric 3D Cad, nave and nave-aisles</td>
<td>PLY v1.0 (binary LE)</td>
<td>3D</td>
<td>Binary (Base64)</td>
<td>Open</td>
<td>—</td>
<td>Embedded</td>
<td>Custom CS1</td>
</tr>
<tr>
<td>4</td>
<td>3D Cad surface, representing damage</td>
<td>OBJ v3.0</td>
<td>3D</td>
<td>Text</td>
<td>Open</td>
<td>—</td>
<td>Embedded</td>
<td>Custom CS1</td>
</tr>
<tr>
<td>5</td>
<td>Detailed survey mesh, nave side-wall Multiresolution mesh, derived from survey</td>
<td>PLY v1.0 (binary LE)</td>
<td>3D</td>
<td>Binary</td>
<td>Multiple textures</td>
<td>—</td>
<td>Referenced</td>
<td>Custom CS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NXZ v4.2</td>
<td>3D</td>
<td>Binary</td>
<td>Proprietary</td>
<td>—</td>
<td>Referenced</td>
<td>Custom CS1</td>
</tr>
<tr>
<td>6</td>
<td>Bounding box, over multiresolution mesh</td>
<td>glTF v2.0 (.gltf)</td>
<td>3D</td>
<td>Text</td>
<td>Open</td>
<td>—</td>
<td>Embedded</td>
<td>Custom CS1</td>
</tr>
</tbody>
</table>
derived SVG plan are included in the graph by connecting the ground floor node to RDF literals referencing the file locations of the plan documents. Similarly as in part 1, the provenance information that the SVG geometry description is derived from the DWG drawing is expressed using the omg:isDerivedFromGeometry property between the SVG and the DWG geometry node. While the DWG drawing is located in a first custom CS, inst:cs1, the SVG derived from the DWG file is located in another named custom CS inst:cs2. In addition, the three Cartesian CSs in use are registered to each other. epsg:6190 is connected to inst:cs1 by way of the inst:transformation1 node, while inst:cs2 is linked to inst:cs1 by way of the inst:transformation2 node. Both transformation nodes are instances of gom:AffineCoordinateSystemTransformation and are linked to their corresponding RDF literal containing a 4 × 4 transformation matrix in a column major array. The units of both custom CSs are metres; thus, no explicit unit has to be modelled in the graph. The SVG geometry description uses pt (point) units internally. Thus, the receiving application has to convert it into metres before doing any transformations between CSs.

Based on the set of existing plan documents from part 2, a volumetric 3D model of the church nave and nave-aisles is created. Each individual construction element of this part (Figure 9(c)) of the building is linked to its corresponding building elements, by linking them to a shared omg:GeometryContext instance. Additionally, they are defined in the same custom CS – that is, inst:cs1.

Part 5 (Figure 10(a)) describes how a large and complex mesh based on a survey is connected to the inst:wallA building element. The large mesh consists of multiple parts – that is, a PLY containing the geometry and several textures. To filter geometry based on file size during querying, the gom:hasFileSize property is added to all geometry nodes of the inspector and the contractor. An alternative, multiresolution representation of the PLY mesh is created in the compressed Nexus format (NXZ) for a better experience inside web applications (Ponchio and Dellepiane, 2016). Both geometry descriptions, located in the previously created custom CS inst:cs1, are stored in external files referenced from RDF literals.

In the final part (Figure 10(b)), the geometry of a relevant area on inst:wallA is created as a bounding volume over the 3D multiresolution mesh geometry of the wall described in part 5. Because a bounding volume is a geometry description on its own, the OMG/FOG linking method can be applied for this case also. The bounding box in the example is stored in the text-based glTF format and subsequently embedded in an RDF literal. They are grouped together with the geometry descriptions of the corresponding building elements, by linking them to a shared omg:GeometryContext instance. Additionally, they are defined in the same custom CS – that is, inst:cs1.

Both omg:Geometry instances, respectively of the wall mesh geometry and the bounding volume, are grouped in an omg:GeometryContext instance. The omg:Geometry node of the bounding volume is also classified using gom:BoundingVolume, allowing the receiving application to select a different rendering style for such geometry. The bounding volume is defined in the custom CS inst:cs1.

In all discussed parts, the OMG/FOG linking method is used, allowing the receiving application to query for the geometry of

Figure 8. Building and damage topology linked with six demonstrative example parts containing geometry descriptions
schemata (including versions) it supports. In data sets of the present case study, exclusively OMG level 2 links (one intermediate omg:Geometry node) are applied. The use of level 1 makes it impossible to add valuable metadata to the geometry description or to group them. Level 3 can be implemented when multiple versions of the same geometry are required – for example, during an iterative design process or in the case of damage geometry that evolves over time. In this study’s example,

Figure 9. Example geometry parts related to the monument inspector: (a) part 1 – inst:church; (b) part 2 – inst:groundfloor; (c) part 3 – inst:wallA and inst:roof1; (d) part 4 – inst:roof1-damage1
data set, GOM is used to deal with the CSs and registrations and to add metadata such as file size and geometry type. Source geometry can be traced back with the omg:isDerivedFromGeometry property, while omg:GeometryContext instances group respective bounding volumes and damage geometry with their parent geometry description.

4.2 Demo web application

In the present case study, the RDF data set of the monument inspector is published using a Triple Pattern Fragments (TPF) server (Verborgh et al., 2016) over an RDF file (Turtle serialisation) hosted on the same server, while the results of the historic research contractor are shared in a single RDF file (Turtle serialisation) stored in an online repository. Both methods result in significantly lower server costs for the publisher compared with a Sparql end point over an RDF database (Verborgh et al., 2016). The TPF server of the inspector is shared with the contractor and the architect, while the link to the Turtle file of the contractor is sent to the architect.

Based on the Comunica framework (Taelman et al., 2018), a wide variety of RDF sources (RDF files, Sparql end points over RDF databases, TPF servers and in-memory RDF stores) can be queried together as one large graph using Sparql. This assures the architect that the most recent published version of each data set is considered during querying. Regular federated Sparql queries can be executed only over multiple Sparql end points, and the boundaries between the considered data sets should be known. Using Comunica, the Sparql SELECT queries from Listings 1 and 2 can be executed over these two RDF sources without reasoning. During the architectural restoration planning, it is even possible to perform combined queries taking a local RDF database of the architect into consideration.

Listing 1. Sparql SELECT query to retrieve each named Cartesian CS and their optional transformations

```sparql
SELECT DISTINCT ?cs ?to ?matrix
WHERE {
?cs a gom:CartesianCoordinateSystem .
OPTIONAL {
 transformation gom:hasMatrix ?matrix ;
 gom:fromCartesianCoordinateSystem ?cs ;
 gom:toCartesianCoordinateSystem ?to .
}
}
```

Listing 2. Sparql SELECT query to retrieve loadable geometry descriptions and their optional named CS

```sparql
SELECT ?elementOrZone ?geometry ?property ?geometryType ?cs ?value
WHERE {
#Retrieve building element or zone presented by the geometry
OPTIONAL {
 ?elementOrZone omg:hasGeometry ?geometry .
}
#Retrieve geometry type if defined
OPTIONAL {
 ?geometry a ?geometryType .
 FILTER (?geometryType != omg:Geometry )
}
```

Figure 10. Example geometry parts related to the historic research contractor: (a) part 5 – inst:wallA; (b) part 6 – inst:wallA-area1
The overall structure of the web application is visualised schematically in Figure 11. The first Sparql SELECT query (Listing 1) retrieves each named CS (?cs) and the optional queried RDF sources.

```sparql
FILTER ( ?BOUND(?size) || ?size < 10000000 ) .
FILTER ( ?property IN (
    fog:asNexus_v4.2-nxz , fog:asGltf_v2.0-gltf ,
    fog:asGltf_v2.0-glb , fog:asPly_v1.0-binaryLE ,
    fog:asPly_v1.0-ascii , fog:asCollada_v1.4.1 ,
    fog:asNexus_v4.2-nxz , fog:asObj_v3.0-obj ,
    fog:asSvg_v1.1 , fog:asPcd_v0.7-ascii ) ) .
```

The second Sparql SELECT query (Listing 2) returns all geometry descriptions (?value), their corresponding geometry schema (?property) and the geometry node (?geometry). In addition, the geometry type (?geometryType), represented building element or zone (?elementOrZone) and optional named CS (?cs) are included.

```sparql
    } .
OPTIONAL { ?geometry gom:hasCoordinateSystem ?cs .
    OPTIONAL { ?geometry gom:hasFileSize ?size .
    FILTER ( ?BOUND(?size) || ?size < 10000000 ) .
    FILTER ( ?property IN (fog:asNexus_v4.2-nxz , fog:asGltf_v2.0-gltf ,
                              fog:asGltf_v2.0-glb , fog:asPly_v1.0-binaryLE ,
                              fog:asPly_v1.0-ascii , fog:asCollada_v1.4.1 ,
                              fog:asNexus_v4.2-nxz , fog:asObj_v3.0-obj ,
                              fog:asSvg_v1.1 , fog:asPcd_v0.7-ascii ) ) .
    }
}
```

The second Sparql SELECT query (Listing 2) returns all geometry descriptions (?value), their corresponding geometry schema (?property) and the geometry node (?geometry). In addition, the geometry type (?geometryType), represented building element or zone (?elementOrZone) and optional named CS (?cs) are included. The query selects geometry descriptions with a combined file size smaller than 100 MB defined in a geometry schema supported by the demo web application. The incoming Comunica result stream in Json is processed automatically by the web application. This includes the fetching of referenced external geometry files, the correct decoding of the text-encoded binary geometry embedded in RDF literals, the management of CSs and calling the correct three.js geometry loader. The URI of each geometry node is added to each loaded three.js geometry object as a reference. Besides the available three.js geometry loaders, the complex meshes in the Nexus format (NXZ) are imported into three.js using a dedicated Nexus loader (Ponchio, 2019). Using the Nexus format, complex meshes are restructured in a multiresolution model, from coarse to detailed. When such a geometry description is imported, it first loads the coarse geometry. If the camera approaches the mesh, the more detailed geometry is loaded. The geometry type is used to render the geometry appropriately – for example, as point cloud, bounding volume or mesh. The application loads the geometry descriptions in their original CS (named or unnamed). If the user activates a certain CS in the application, all geometry already defined in this CS is rendered. Geometry descriptions in other CSs that are registered to the active CS are transformed using Equation 1 (affine transformations) and subsequently rendered. Figure 12 contains the rendering in the web application (inst:cs1 as active CS) of the loaded geometry from the two data sets, located either in inst:cs1 or in a CS registered to inst:cs1, combined with the related graph nodes.

## 5. Discussion

Different pieces of geometry – 2D and 3D, binary or text-encoded, in open or proprietary schemas – are created in different applications and have to be exchanged in a controlled manner with other built heritage stakeholders to ensure an efficient collaboration. To avoid conversions between the original, widespread geometry schemas and complex RDF-based geometry schemas with limited support in existing applications, the method described in Section 3 is applied. By using RDF literals to either embed or reference geometry descriptions, any existing geometry...
schema and thus any geometry-enabled application can be used in a Linked Data context. The exchanged geometry in the case study mainly includes mesh, point cloud and plan geometry, but the same method can be used for the exchange of editable geometry descriptions in schemas such as Step, Rhino and Revit (Bonduel et al., 2019). Both embedding and referencing geometry descriptions can occur in one data set. If the geometry descriptions are referenced files in RDF literals, the access to these files has to be ensured for each receiving stakeholder. When geometry descriptions are embedded in RDF literals, a limited amount of preprocessing is needed – that is, text-encoding in Unicode characters in the case of a binary schema and proper character escaping for text-based geometry schemas.

Using the OMG/FOG ontologies and the related modelling method, the separate concepts of geometry description and semantic description can be uniformly connected in a Linked Data context. A certain object can be linked to one or multiple geometry descriptions consisting of multiple pieces – for example, a PLY mesh geometry accompanied by several textures. Additional metadata such as CSs and transformations, length units, number of vertices/faces/edges, file size, author and creation date can also be added to geometry descriptions in OMG level 2 or 3 using terminology from the GOM ontology. Geometry grouping and derivation relations can be added using the corresponding OMG terminology. OMG level 3 patterns are not considered, as the case study does not handle an evolving restoration design or damage propagation over time. Another relevant topic is spatial querying in Sparql. Existing extensions of Sparql consider only specific geometry schemas and linking patterns. Examples are GeoSparql and stSparql for 2D WKT and GML (Kyzirakos et al., 2012) and BimSparql for 3D WKT (Zhang et al., 2018). At the moment, there exists no Sparql extension that recognizes the OMG/FOG linking patterns to link WKT/GML geometry or geometry described in other geometry schemas.

All documented HBIM workflows applying Linked Data described in Section 2 use conventional BIM authoring tools to model the building semantics and geometry, before these are converted into applications to request only the geometry that they can load. The envisaged modelling approach allows applications to process disparate geometry descriptions automatically, as demonstrated in the demo web application. While three OMG levels of complexity are defined, mainly levels 2 and 3 are considered applicable for collaborative built heritage projects. These two complexity levels allow linking an object to several different geometry descriptions consisting of multiple pieces – for example, a PLY mesh geometry accompanied by several textures. Additional metadata such as CSs and transformations, length units, number of vertices/faces/edges, file size, author and creation date can also be added to geometry descriptions in OMG level 2 or 3 using terminology from the GOM ontology. Geometry grouping and derivation relations can be added using the corresponding OMG terminology. OMG level 3 patterns are not considered, as the case study does not handle an evolving restoration design or damage propagation over time. Another relevant topic is spatial querying in Sparql. Existing extensions of Sparql consider only specific geometry schemas and linking patterns. Examples are GeoSparql and stSparql for 2D WKT and GML (Kyzirakos et al., 2012) and BimSparql for 3D WKT (Zhang et al., 2018). At the moment, there exists no Sparql extension that recognizes the OMG/FOG linking patterns to link WKT/GML geometry or geometry described in other geometry schemas.

All documented HBIM workflows applying Linked Data described in Section 2 use conventional BIM authoring tools to model the building semantics and geometry, before these are converted into...
Linked Data. While such conventional BIM tools are well suited for the modelling of regular new buildings, it is not always straightforward to use them for the modelling of heritage objects. Additionally, based on a stakeholder analysis in the Flemish heritage sector, few organisations actually use conventional BIM applications. As a matter of fact, the semantic building modelling can also be implemented directly in Linked Data – for example, using the proof-of-concept web application described by Bonduel et al. (2018b). By avoiding conventional BIM applications altogether, a challenging conversion from a conventional BIM schema (IFC or proprietary) to a Linked Data schema for buildings can be avoided. In the workflow proposed in this paper, stakeholders are enabled to apply the BIM concept by performing the semantic modelling directly in Linked Data, while the geometry modelling is realised in a separate Cad or reality capturing application. In the final output, nodes in RDF describing objects and the corresponding geometry descriptions stored in or referenced from RDF literals are connected using the OMG/FOG terminology. The scope of this paper is limited to the necessary RDF graph structures and a demo visualisation application. Before these RDF patterns can be applied in practice, however, software developers need to design more intuitive user interfaces for the creation of HBIM graphs by end users.

6. Conclusion

The presented work discussed the need for the inclusion of widespread geometry schemas in RDF graphs describing built heritage buildings. RDF literals were successfully applied to embed or reference a wide variety of geometry descriptions, which is in fact useful for many construction projects using Linked Data. A flexible and uniform linking method was applied in combination with the OMG ontology module containing the related terminology. The compatible terminology modules FOG and GOM were developed to make the reuse of geometry in a Linked Data context more convenient. According to best practices, these ontologies are made available online in their own persistent namespace and indexed in the Linked Open Vocabularies repository (Vandenbussche et al., 2017). Sample RDF triples and example queries are included in an online Sparql-visualizer demo. The effectiveness of the entire data structure is demonstrated in a theoretic built heritage case study project involving an architect, a monument inspector and a heritage research contractor. A variety of geometry – including point clouds, 3D survey meshes, 2D plans and 3D Cad geometry – describing a medieval church in Ghent is assembled and included in a Linked Data context. The architect who receives the RDF graphs from the other stakeholders is able to create successfully a coordinated view using the demo web application on geometry descriptions in shared, but disparate RDF data sets. While the theoretic workflow responds to real industry needs, the proposed data structure needs to be tested and evaluated more thoroughly by domain experts in the future. To increase the uptake of the proposed domain-agnostic data structure in a broad range of software implementations and data sets, outreach to the W3C Spatial Data on the Web Interest Group and other standardisation bodies will be considered. As described in the discussion, no spatial querying extension for Sparql currently exists that considers other geometry schemas than WKT and GML and recognises the OMG/FOG linking patterns. It is impossible to create a Sparql extension that can deal with geometry of every existing geometry schema, simply because there is no geometry kernel that can deal with every schema. In the future, a series of Sparql extensions can be created that recognise the OMG/FOG linking patterns and, depending on the implemented geometry kernel, support a selection of geometry schemas.

Acknowledgements

This research is funded by the Research Foundation Flanders in the form of a personal Strategic Basic research grant (grant agreement 1S65917N). The authors would like to thank the participants of the stakeholder analysis and specifically the Facility Management Department of the City of Ghent for sharing 3D Cad geometry representing the Saint Nicolas Church. The Geomatics Research Group of the Katholieke Universiteit Leuven Department of Civil Engineering is gratefully acknowledged for delivering a detailed survey mesh of the case study church.

REFERENCES

De Luca L, Busayarat C, Stefani C, Véron P and Florenzano M (2011) A semantic-based platform for the digital analysis of architectural...
including widespread geometry schemas into Linked Data-based BIM applied to built heritage

Bonduel, Wagner, Pauwels, Vergauwen and Klein


Sikos LF (2017) A novel ontology for 3D semantics: ontology-based 3D model indexing and content-based video retrieval applied to the

How can you contribute?
To discuss this paper, please email up to 500 words to the editor at journals@ice.org.uk. Your contribution will be forwarded to the author(s) for a reply and, if considered appropriate by the editorial board, it will be published as discussion in a future issue of the journal.

Proceedings journals rely entirely on contributions from the civil engineering profession (and allied disciplines). Information about how to submit your paper online is available at www.icevirtuallibrary.com/page/authors, where you will also find detailed author guidelines.