bcfOWL: A BIM collaboration ontology

Oliver Schulz[1,2], Jyrki Oraskari[1,3][0000−0002−4723−3878], and Jakob Beetz[1,4]

[1] Department of Design Computation, RWTH Aachen University, Aachen, Germany
[2] schulz@dc.rwth-aachen.de
[3] Jyrki.Oraskari@dc.rwth-aachen.de
[4] j.beetz@caad.arch.rwth-aachen.de

Abstract. The BIM Collaboration Format (BCF) is a buildingSMART standard used to exchange issues in a digital model between heterogeneous software applications and planners. Although the BCF issues are spatially located in a model and provide links to building elements, the format is only loosely connected to the actual BIM model. While the format is well suited for its original use case, it lacks the flexibility to retrieve information from the BCF in conjunction with the BIM model. In this paper, we introduce the BIM Collaboration Format Ontology (*bcfOWL*), which translates the format to the Semantic Web and harnesses the expressive richness of OWL. We present the structure of the ontology and highlight the differences with existing implementations of BCF. Using example queries, we show how extended relationships between a BIM model and BCF information are enabled by *bcfOWL*. We show that by transferring BCF to Linked Data, the format's applicability is enhanced without losing compatibility with existing implementations and workflows. In doing so, the ontology aims to integrate into the Linked Building Data environment and facilitates access to synergies between heterogeneous building data.

Keywords: BCF · SPARQL · OWL · REST API · XML · Linked Data

1 Introduction

Issue management is an integral part of most BIM processes in the Architecture, Construction, Engineering, Owner and Operator (AECOO) industry. Today this is often implemented by using the BIM Collaboration Format (BCF) of the buildingSMART organization. It represents a standardized way of communicating issues between heterogeneous software applications. There are two standard approaches for exchanging data: a file-based workflow, where *Issues* are written into a ZIP container using the BCF XML[5] schema, and a server-based approach, which uses the BCF API[6] specification for the data exchange. It is based on the principles of a RESTful API. It allows users of different disciplines using different software to communicate the reported *Issues* online, eliminating the need to exchange files via email or a data device. [4]

[5] https://github.com/buildingSMART/BCF-XML
[6] https://github.com/buildingSMART/BCF-API

Common use cases for such *Issues* include clash detection during partial domain model integration and change or information request in the design phase of a building. They often result in hundreds of *Issues* in a single information handover or integration scenario. For such common scenarios, the management of *Issues* captured in individual files is highly impractical and queries for specific data are challenging to implement via the API.

Over the last years, there have been many approaches to compose Linked Building Data. The ifcOWL ontology [3, 10] was designed to be as equivalent as possible a Semantic Web representation of the IFC EXPRESS schema. IfcWoD was designed to express the object-oriented constraints of the IFC schema as an OWL ontology [6]. BimSPARQL [17] proposed an extension for SPARQL Query Language (SPARQL) to query geometrical and BIM specific data of IFC models. Like ifcWoD and BimSPARQL, SimpleBIM was an attempt to create a more accessible view on ifcOWL data. [9]. Recently, the evolution culminated in the development of the Building Topology Ontology (BOT) [13], which reduced the extensiveness compared to ifcOWL and enabled modular ontology design when describing buildings in the Linked Data context.

The Linked Building Data Community Group (LBD-CG)[7] of the World Wide Web Consortium (W3C) is devoted to advance the adoption of the Linked Data and Semantic Web technologies in AECOO. The vision is to enable web-based information exchange and workflows, provide open and versatile standards for different domains, and facilitate distributed data integration. [13]

In this paper, we present the BIM Collaboration Format Ontology (*bcfOWL*)[8] that shares the principles of Linked Building Data, and which provides a way to create BCF information that is interlinked in the context of a building. We describe the design principles used and demonstrate the benefits of the approach by example queries. We discuss the differences from the BCF API server definition and argue how the representation of *Issues* as semantic graphs are beneficial in continuous information integration scenarios in common BIM-based workflows. The underlying research questions are:

1. How can *Issues* that arise in common workflows be integrated and linked with the BIM workflow?
2. How to express the core ideas of BCF by following the best practices of ontology design and still keeping the result compatible with the original BCF API and BCF XML standards?

The paper is structured as follows: The first section describes the current implementations of BCF and highlights a first approach of converting BCF to an ontology. The following section deals with the newly created ontology, *bcfOWL*, and focuses on which design principles and best practices were considered when creating it, and how it compares to the current implementations of BCF. In section 4, we provide example use-cases and show how SPARQL can be used to query BCF information. We discuss the results of *bcfOWL*, conclude the paper, and give an overview of future work in section 6.

[7] `https://www.w3.org/community/lbd/`
[8] `http://lbd.arch.rwth-aachen.de/bcfOWL#`

2 BIM collaboration Formats

2.1 BCF XML

BCF was initially developed by the companies Solibri and Tekla, and the Institute of Applied Building Informatics to exchange problems within a digital building model between different software applications.[9] The format is often used in the area of issue management and collision checking. The original version of BCF is based on an XML format, and the *Issues* are organized in sub-folders within a ZIP file. This file can then be distributed and imported into other software applications that support the standard. The BCF XML format also allows round-tripping, where the same file can be extended with additional *Issues*, and existing *Issues* can be modified. Thereby preventing that the *Issues* are stored over several files, and thus a clear structure is lost.

2.2 BCF API

Although this round-tripping of the BCF XML files already reduces ambiguity and helps stricter, machine-readable representations of *Issues*, the exchange of files via e-mail or data storage media is a weak point of related processes and workflows. [4] To address these shortcomings, the BCF API was developed to replace the file-based exchange of *Issues* in the digital building. This is achieved through a server infrastructure that enables communication via a RESTful API with client applications. Rather than in XML, the data is captured and processed using the JSON format. Providers of BCF servers enable the retrieval, viewing and editing of *Issues* by external software applications or on websites, which allows a wide range of stakeholders within a project to be integrated into the planning process.

2.3 Common Data Environments

The 2.1 version of the BCF API contains a *Public Service*, responsible for a rudimentary *User Management*, and a *Documents Service*, which can be used to interchange BIM models via the BCF server and link an *Issue* to a specific *Document* or *File*. Version 3.0 already separates the *Public Services* to an API, called the Foundation API and serves as a common base layer on an upcoming multi-tier architecture for open standards for Common Data Environments[10] (openCDE APIs). For the *Document Service*, a Documents API is under development as well. These APIs are intended to enable a standardized, vendor-independent exchange of BIM models and their associated data. BCF is an integral part of this infrastructure, as it can be used within an openCDE to document issues and create change requests across multiple documents.

[9] `https://technical.buildingsmart.org/standards/bcf/`
[10] `https://technical.buildingsmart.org/projects/opencde-api/`

2.4 General structure of BCF

Although BCF XML and BCF API are based on different data serialization formats, both variants are compatible on a structural level. They are based on the basic concepts of *Projects*, *Topics*, *Comments*, and *Viewpoints*, which are organized hierarchically. The combination of *Topics*, *Viewpoints*, and *Comments* is often referred to as an *Issue*. When talking about an Issue, this is to be equivalent to the Topic, which is the connecting element for the Viewpoints and Comments. In the file structure of the BCF XML, the term *Markup* is used. The general structure can be reviewed in Figure 1.

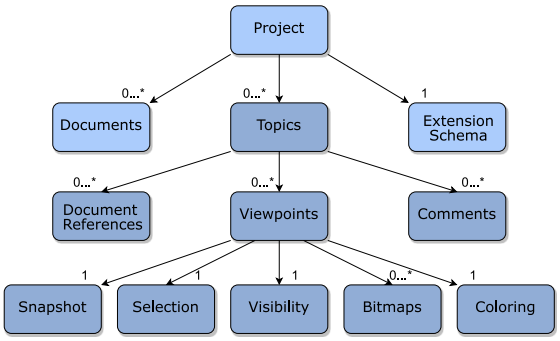

Fig. 1. General structure of the BCF. The dark blue boxes form the *Issue*. Each hierarchy level represents a request to the server in the BCF API.

Except for free text parameters, the *Topics* must be filled with data defined in the *Extension Schema*. The concept of *Viewpoints* establishes a spatial link to the BIM model and conveys the information about which parts of the building are visible or selected in the respective camera view. The position and rotations are specified in a Cartesian coordinate system with vectors (X—Y—Z).

2.5 Drawbacks of current BCF implementations

In [14], the hierarchy was identified as one of the key problems of the BCF API since the nested structure demands a specific way of requesting information. If e.g. all *Comments* created by a particular person are requested from the server, all Topics must be loaded first, and their IDs can then be used to request the *Comments* associated with the *Topics*. Thus an overhead of requests to a server is created. The same matter applies to the *Viewpoints*, which contain sub-concepts that have to be requested from the server separately. This is a common problem when working with REST APIs and is regarded as the "N+1 problem"[11].

Furthermore, the connection to BIM happens via two parameters in the BCF Format. On the one hand, the *Issues* are spatially linked with the model via

[11] `https://restfulapi.net/rest-api-n-1-problem/`

location and rotation in the *Viewpoints*. On the other hand, the *Components* subconcept can link to a building element via Globally Unique Identifier (GUID). Nevertheless, currently neither allows to query BCF information in a spatial context or a context of the accompanying building elements referenced by the *Issue*. Already downloaded *Issues* and BCF XML could be queried with custom-designed functions in such a way, but just in the context of the BCF information currently loaded into the model.

2.6 BCF in Linked Data

A first approach of mapping the BCF format to Linked Data can be seen in [5] where a BCF ontology was created by converting the XSD schema of BCF XML, using the XsdImport plugin for TopBraid Composer. The created ontology is very close to the file-based principles of BCF XML. It thereby uses concepts like the *Markup* and multiple properties that are redundant for an online distributed approach and add to the complexity of the ontology. Moreover, most ontology properties are converted into Datatype Properties and used as a string. Cardinality constraints for the properties are not provided for the ontology classes. Furthermore, the ontology was not published, and it remained as a test use case in a larger prototype.

3 bcfOWL

bcfOWL aims to lift BCF to the Linked Data domain and thereby enable the format to be put in a larger context within the Linked Building Data, including the BOT and ifcOWL ontology. The conversion should not just be a translation of the BCF API or the BCF XML but specifically tailored to the needs in a Linked Data context. At the same time, compatibility with existing formats should not be jeopardized. Therefore we considered guides, checkers, and best practices to achieve this goal:

- In [7], the author argues that the complexity of classes inside an ontology should be kept simple for the ontology to be usable by implementers and system developers. This is exemplified by Container classes that hold no actual data and thus should be avoided when designing an ontology.
- In [12], the authors define good and bad practices, referred to as pitfalls when publishing vocabulary on the web and provide detection methods for these practices.
- In [11], an online tool is provided to scan ontologies for these pitfalls, which was used to validate *bcfOWL*.
- Moreover, best practices were also defined by the World Wide Web Consortium[12] (W3C) that were also put into consideration when designing *bcfOWL*.

[12] https://www.w3.org/TR/ld-bp/

Since the work aims to translate the BCF format into an ontology, the main difference between the two formats is the description in the respective schema. However, a notable characteristic is that BCF exists in both a file-based and a server-based approach (Section 2). The *bcfOWL* approach exists between these two and exploits both variants.

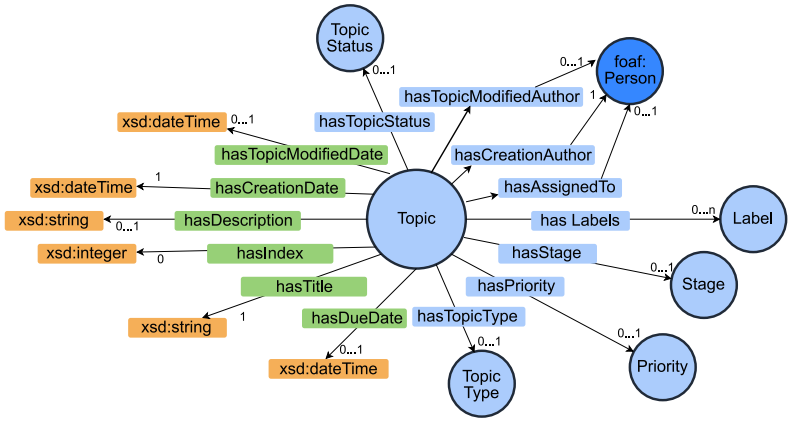

Fig. 2. A snippet from the ontology, displaying the *Topic* as a graph

bcfOWL is covering the core abstraction and concepts of the BCF standard relating the Issue Management. The subsets of *Event Services*, *File Services*, and *Document Services* are not included into this core ontology. Supplementary transfers of these modules into ontologies would also have to be provided and form a Linked Data Common Data Environment, together with *bcfOWL*.

3.1 Projects

The *Projects* are a core element of BCF responsible for assigning each *Issue* to a project, and thus a specific building. The link to the *Project* in the BCF API happens via URL (Listing 1.1), where the *Projects* GUID is provided. In BCF-XML, the *Project* is defined per file and there is no further assignment to it. Since resources from *bcfOWL* are not queried via parameters in the URL, and a file-based approach is not the sole scope of our work, the link to the *Project* has to be provided in the classes of the *Topics*, *Comments*, and *Viewpoints*. (Listing 1.2) The *Project* itself links to the *Extension Schema*, described in the following section.

```
GET /bcf/3.0/projects/{project_id}/topics
```

Listing 1.1. GET request to a BCF server. The *Project* is provided as parameter

```
@prefix bcfOWL: <http://lbd.arch.rwth-aachen.de/bcfOWL/> .

bcfOWL:Comment rdf:type owl:Class ;
  rdfs:subClassOf
    [ rdf:type owl:Restriction ;
      owl:onProperty bcfOWL:hasProject ;
      owl:cardinality "1"^^xsd:int ] ,
    [...] .
```

Listing 1.2. The *Project* has to be provided for the *Topics*, *Viewpoints*, and *Comments* and therefore a cardinality of 1 is specified.

3.2 Extensions

In the BCF context, the *Extensions* are used to create project-specific parameters when inserting new *Issues*. In a *Project*, parameters such as *Status*, *Labels*, and *Users* are defined in advance in the *Extensions*. With parameters such as *Title* or *Comment*, on the other hand, free texts are used. Especially with the BCF API, the correct parameters are controlled by the server during the upload since it only allows the parameters mapped in the *Project's Extensions*.

bcfOWL also provides the possibility to define these *Extensions* to be used in the *Projects*. However, since the allowed values can change from project to project, we provide only a generic and backwards-compatible way to define the property sets. The extension and validation mechanisms will be studied further in future work.

3.3 Topics, Comments, and Viewpoint

Same as in the *Projects*, the *Topics*, *Comments*, and *Viewpoints* need to be linked to each other to generate a complete *Issue*. In the BCF API, these links are primarily provided in the URL parameters. For example, the link of the *Viewpoint* to its *Topic* is only provided by the URL and is not part of the JSON schema. In the BCF XML, this link is provided by the *Markup*. In *bcfOWL*, these connections are provided in the respective classes of the *Comments* and *Viewpoints* as Object Properties, linking to the *Topic*. The *Markup* concept was discarded, thus reducing the complexity of the ontology. This is in line with BCF API, where it had been left out.

3.4 Issue transformation

The transformation of an *Issue* is an integral part of the principles of the BCF. It is described with a *Camera View Point* in a Cartesian coordinate system, a *Camera Direction*, an *Camera Up Vector* for describing the rotation of the view in the coordinate system, and an *Aspect Ratio*. Depending on if the camera is described as an *Orthogonal Camera* or a *Perspective Camera*, a *View to World-Scale* or a *Field Of View* has to be provided. The transformation of the camera is described in the *Viewpoint* of an *Issue*. To express the vectors and points,

bcfOWL relies on the representation in well-known-text (WKT) as proposed for the ifcOWL format in [8]. Using WKT, compatibility with GeoSPARQL [2] should be achieved, a standard of the Open Geospatial Consortium (OGC). *bcfOWL* is also inspired by BimSPARQL [17], which describes points with the WKT. It is an extension of the SPARQL query language, with a specific focus on BIM and ifcOWL. Both query languages allow to query data in a spatial context from the graph and apply constraints to the query like a maximum distance between two points [17].

3.5 Snapshots

In BCF, most *Issues* are accompanied by a *Snapshot* from the source application to help understand the problem described since not all models are loaded in every software in which an *Issue* is considered. Thus, the problem cannot be verified without further context. In the BCF XML, the images are stored as PNG or JPEG in the sub-folder with the *Markup*. They are linked to the *Viewpoint* via an GUID. In the BCF API, when a new *Snapshot* is created, the image is passed in base64 format, included in a JSON. When requesting a *Snapshot*, the image is passed as a binary file from the BCF server. Since including images with base64 strings in RDF leads to extensive and slow graphs, the locations of the images are linked in *bcfOWL* by URI. For compatibility reasons, the URI is the same as in the BCF API approach and the up- and download of the *Snapshots* can be achieved by simple GET and POST requests.

4 Example use-case

By introducing BCF to OWL, we can now use this format with more advanced queries, where the user can specify what data should be fetched from a server. In this section, we provide an example use case for *bcfOWL*. The queries were tested with Apache Jena Fuseki.

4.1 Querying BCF information

While a request with the BCF API, for example, for a *Topic*, can be handled via a simple GET request (Listing 1.1), the same request in the SPARQL context is more extensive (Listing 1.3). However, this problem can be partially circumvented by using template functions to map the standard requests to the BCF server used in the REST API.

```
PREFIX bcfOWL: <http://lbd.arch.rwth-aachen.de/bcfOWL/>

SELECT ?s ?p ?o
WHERE {
    ?s a bcfOWL:Topic ;
      ?p ?o .
}
```

Listing 1.3. Sample SPARQL query for getting *Topics*.

4.2 Query by Selection

An important feature to bring building data closer together with BCF is to query building elements to see if *Issues* related to them already exist. Software applications such as Solibri discovered the need for such information and have integrated a corresponding feature into their application. However, the feature only supports a local (offline) context and cannot align the *Issues* for the building elements with the *Issues* stored on a server without downloading them first. Specific queries like these are not possible with the current API, or only to a limited extent, because the existing hierarchy requires all *Issues* on the server to be searched first to get the IDs of the building elements. [14] However, by implementing the *bcfOWL* ontology it is now possible with a single request. By providing the ID of a selected building element, an application can send a query to the database with the searched for GUID and the server responses with all *Topics* referring to the GUID.

```
PREFIX bcfOWL: <http://lbd.arch.rwth-aachen.de/bcfOWL/>

SELECT ?topic
WHERE {
  ?topic a bcfOWL:Topic .
  ?viewpoint a bcfOWL:Viewpoint ;
           bcfOWL:hasTopic ?topic ;
           bcfOWL:hasSelection ?selection .
  ?selection bcfOWL:hasComponent ?component .
  ?component bcfOWL:hasIfcGuid "1ZwJH$85D3YQG5AK5ER1Wc" .
}
```

Listing 1.4. SPARQL example for querying *Topics* with a specific GUID in the selection.

4.3 Referencing Building Elements

Referencing a Building Element in the *Selection* or the *Visibility* can either be achieved by a GUID, an Authoring Tool ID, or by linking an object (by a dereferenceable URL). The first two options seem to be less in line with Semantic Web concepts but is helpful if the IFC and its elements do not exist in a Linked Data context. This ensures compatibility with a wide range of formats while linking with a Linked Data object is still possible.

```
inst:Element_1 a bcfOWL:Component;
   bcfOWL:hasIfcElement inst:BotElement_1;
   bcfOWL:hasIfcGuid "1zMXMGg3H1MOhxQ9qjUS4B" ;
   bcfOWL:hasAuthoringToolId "312213" ;
   bcfOWL:hasOriginatingSystem inst:ExampleCAD .
```

Listing 1.5. Components provide a link and/or the GUID of the element.

5 Discussion

The examples show that *bcfOWL* can address use cases beyond the capabilities of the current BCF formats. It was demonstrated by the example of querying a graph database for building elements for which *Issues* exist. The connection to the building elements can be described with the ontologies of BOT and ifcOWL via URIs and a conventional IFC or native BIM model via GUID. *bcfOWL* can thus be integrated into existing workflows. Although individual parts have been excluded from *bcfOWL* , the underlying concepts remains the same. A conversion of *bcfOWL* data to BCF XML or BCF API is still applicable.

However, this extended functionality comes at the cost of using the SPARQL language, which can be a hurdle for users and developers [15]. Especially when compared to requests to a REST service, requests via SPARQL become more complex and require a deep understanding of the format to properly represent BCF in a graph while maintaining compatibility with the existing format. By the absence of validation of newly created resources, which can cause data to be uploaded incorrectly in the context of the BCF format, this is even intensified. Furthermore, the GeoSPARQL ontology was used for describing the position of the Issues in a three-dimensional context with WKT. However, GeoSPARQL in its current version is mainly limited to a spatial context in a 2D domain. Although three-dimensional vectors can be represented in the WKT system, only the XY coordinates are checked for spatial queries to the graphs. Nevertheless, use cases for three-dimensional queries have been recognized [1] by the OGC and have been included as a milestone[13] in the further development of the format. Because of the limitation no spatial queries were tested for *bcfOWL*.

Since the concept of inverse relationships does not exist in BCF, it has not been incorporated into the ontology at this time. It should be discussed in the future whether the advantages of inverse relations justify a more complex ontology.

6 Conclusion and future work

We have shown in this paper that a transformation of BCF into a Linked Data approach is viable and opens new possibilities for the format. While BCF XML and BCF API applications can actualize and interpret the indirect GUID references of a BIM model when the BIM models are loaded to the memory, the proposed Linked Data approach allows better integration. The new data format makes it easier to retrieve relevant issue data and attach and track provenance information over different life-cycle stages. Further, deeper analysis is left for future work. Using Linked Data, the BCF and BIM data models can be integrated into a shared knowledge graph, making it possible to harness the power of semantic queries, e.g. using a SPARQL engine. Spatial localization is achieved via the WKT format, but further emphasis needs to be placed on enabling and testing spatial queries.

[13] `https://github.com/opengeospatial/ogc-geosparql/issues/19`

bcfOWL is by no means intended to replace the existing formats but merely to add further possibilities. A conceivable scenario would be to use *bcfOWL* as an internal data format for a BCF server, which can then be exposed via a SPARQL endpoint, and via an intermediary service using the BCF API. Thus the strengths and weaknesses of the respective formats could be balanced, and compatibility to already existing approaches could remain. The proposed method allows using Linked Building Data content like assertions that use ifcOWL or the BOT ontologies connected with or beside regular IFC and BIM models. The data can also be linked to available Linked Data sources.

When creating *bcfOWL*, we focused on the core functions necessary when creating *Issues* within a building. Subjects such as handling and linking to and from *Documents* and *Files* and user management were excluded from the *bcfOWL* definition. Like bundling the different buildingSMART APIs, a Linked Data based Open CDE infrastructure could also be applied here. First approaches to such considerations can already be observed in [16]. Also, the tracking of the BCF *events* is not integrated into *bcfOWL*, and further work should show how these can be used in a meaningful way in the Linked Data context. BCF *Events* have an essential role in Issue management, as they can be used to track who changed what data in a BCF and when.

Another starting point in conjunction with *bcfOWL* could be a connection to the Shapes Constraint Language[14] (SHACL), which is used to validate RDF graphs. Since in the BCF context, *Projects* are provided with *Extensions*, and thus no parameters should be used in these *Projects* that do not appear in them, a SHACL implementation would be a potential complement to *bcfOWL*.

By introducing BCF into the Semantic Web, we hope that the application possibilities of the format will be extended. This includes increased possibilities to analyze BIM processes and harness the underlying data graphs to interlink and trace dependencies, provenance, and historical evolvement of *Issues*. We also see potentials in the interoperable interlinking of *Issues* with heterogeneous, non-IFC, and legacy information types for multi-model containers, e.g., the ICDD and OpenCDE-APIs. *bcfOWL* should enable synergies with other ontologies such as ifcOWL and the Building Topology Ontology and thus contribute to the goal of Linked Building Data.

7 Acknowledgments

This research had been funded by the EU through the H2020 project BIM4REN.

References

1. Abhayaratna, J., Brink, L., Car, N., Atkinson, R., Homburg, T., Knibbe, F., Mcglinn, D.K., Wagner, A., Bonduel, M., Rasmussen, M.H., Thiery, F.: OGC Benefits of Representing Spatial Data Using Semantic and Graph Technologies. Tech. rep., Open Geospatial Consortium (Oct 2020)

[14] `https://www.w3.org/TR/shacl/`

2. Battle, R., Kolas, D.: Geosparql: enabling a geospatial semantic web. Semantic Web Journal **3**(4), 355–370 (2011)
3. Beetz, J., van Leeuwen, J., Vries, d.: IfcOWL: A case of transforming EXPRESS schemas into ontologies. AI EDAM **23**, 89–101 (Feb 2009). https://doi.org/10.1017/S0890060409000122
4. van Berlo, L., Krijnen, T.: Using the bim collaboration format in a server based workflow. Procedia Environmental Sciences **22**, 325–332 (2014)
5. Bernal Ferrer, F.: Internal design validation attestation in BIM models: a combination of semantic web technologies and BIM collaboration format. Master's thesis, Eindhoven University of Technology, Eindhoven (Oct 2017)
6. Mendes de Farias, T., Roxin, A.M., Nicolle, C.: IfcWoD, Semantically Adapting IFC Model Relations into OWL Properties. In: Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands. pp. 175–185 (2015)
7. Hammar, K.: Ontology design principles for model-driven applications. In: Advances in Pattern-Based Ontology Engineering, pp. 273–278. IOS Press (2021)
8. Pauwels, P., Krijnen, T., Terkaj, W., Beetz, J.: Enhancing the ifcOWL ontology with an alternative representation for geometric data. AUTOMATION IN CONSTRUCTION **80**, 77–94 (2017). https://doi.org/10.1016/j.autcon.2017.03.001, publisher: ELSEVIER
9. Pauwels, P., Roxin, A.: SimpleBIM : From full ifcOWL graphs to simplified building graphs. In: Christodoulou, S., Scherer, R. (eds.) eWork and eBusiness in Architecture, Engineering and Construction (ECPPM). pp. 11–18. CRC Press, Limassol, Cyprus (2016)
10. Pauwels, P., Terkaj, W.: Express to owl for construction industry: Towards a recommendable and usable ifcowl ontology. Automation in Construction **63**, 100–133 (2016)
11. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. International Journal on Semantic Web and Information Systems (IJSWIS) **10**(2), 7–34 (2014)
12. Poveda-Villalón, M., Vatant, B., del Carmen Suárez-Figueroa, M., Gómez-Pérez, A.: Detecting good practices and pitfalls when publishing vocabularies on the web. In: WOP (2013)
13. Rasmussen, M.H., Lefrançois, M., Schneider, G., Pauwels, P.: Bot: the building topology ontology of the w3c linked building data group. Semantic Web (11 2020). https://doi.org/10.3233/SW-200385
14. Schulz, O., Beetz, J.: Image-documentation of existing buildings using a server-based bim collaboration format workflow. In: Abualdenien, J., Borrmann, A., Ungureanu, L.C., Hartmann, T. (eds.) EG-ICE 2021 Workshop on Intelligent Computing in Engineering. pp. 108–107 (2021)
15. Werbrouck, J., Senthilvel, M., Beetz, J., Pauwels, P.: Querying heterogeneous linked building datasets with context-expanded graphql queries. In: 7th Linked Data in Architecture and Construction Workshop. vol. 2389, pp. 21–34 (2019)
16. Werbrouck, J., Taelman, R., Verborgh, R., Pauwels, P., Beetz, J., Mannens, E.: Pattern-based access control in a decentralised collaboration environment. In: Proceedings of the 8th Linked Data in Architecture and Construction Workshop. CEUR-WS. org (2020)
17. Zhang, C., Beetz, J., de Vries, B.: BimSPARQL: Domain-specific functional SPARQL extensions for querying RDF building data. Semantic Web **9**(6), 829–855 (2018). https://doi.org/10.3233/SW-180297