SHACL is for LBD what mvdXML is for IFC

Jyrki Oraskari, Jyrki.Oraskari@dc.rwth-aachen.de
Department of Design Computation, RWTH Aachen University, Aachen, Germany

Madhumitha Senthilvel, senthilvel@dc.rwth-aachen.de
Department of Design Computation, RWTH Aachen University, Aachen, Germany

Jakob Beetz, j.beetz@caad.arch.rwth-aachen.de
Department of Design Computation, RWTH Aachen University, Aachen, Germany

Abstract
The motivation for this work stems from an EU – funded project, which focuses on leveraging digital tools for improving the renovation processes. In particular, specific tools require Linked Building Data (LBD) that need to fulfil the application-specific exchange requirements. In this research, we focus on two different use cases to investigate how to validate a Linked Building Data model. First, we study how to minimise data loss and errors when data is converted and brought into an LBD data store. The usage of unit tests to improve conversion quality is introduced. The second use case focuses on how Model View Definition (MVD) in LBD for evaluating the energy performance of the renovation designs in energy simulation can be formed. This feasibility study shows that unit test can be written the conversion. Besides validation, methods shown in the study can be used to create model views for LBD data using SHACL.

Keywords: IFC. MVD, LBD, SHACL, OWL, RDF, unit test paper

1 Introduction

Information management of the stakeholders' data is one of the critical challenges in the construction industry. Linked Data is an approach that has been proposed to lower the barriers of data silos by using web technologies, Resource Description Framework (RDF), and recommended best practices for expressing and sharing information.

The Linked Data for architecture, engineering and construction (AEC) applies the relevant vocabularies and structured interlinked data to publish content in a machine-interpretable form that can be used for semantic queries.

The new model presentation can be used for model checking, machine inferencing, lossless data exchange, and an extendable interlingua for the AEC programs, i.e., supporting interoperability between software applications. (Pauwels, et al., 2017)

The motivation for this work stems from an EU – funded project which focuses on leveraging digital tools for improving the renovation processes. As part of this project, various workflows for chaining tools used for common renovation-specific use cases were developed. It was identified that the tools in such a workflow need to be supplied with the appropriate data necessary for them to function seamlessly. In particular, certain tools required the use of Linked Building Data (LBD) inputs, which were arrived at from IFC files through the usage of BOT-based converters.

This research focuses on two different use cases to investigate how to validate a Linked Building Data model. First, we study how to minimise data loss and errors when data is converted into LBD and how unit tests can improve the Linked Data quality when it is bought into the

system. The second use case focuses on how Model View Definition in the context of the LBD data for evaluating the energy performance of the renovation designs in energy simulation can be formed.

There have been studies on validating LBD data earlier. Here we focus on how to create a validated MVD view of LBD data using SHACL.

In the following chapters, we first review the Linked Data in the AEC domain, how the data has been converted. Shapes Constraint Language (SHACL) is presented and how it has been applied to LBD data of validation. In Chapters 3 and 4, we introduce our selected use cases, and define the criteria for the validation rules in the context. How the requirements can be tested using SHACL are analysed. Finally, the findings are discussed, and conclusions are summarised.

2 Related Work

2.1 Linked Building Data

Over the last decade and a half, several linked building data ontologies have been proposed in the Architecture, Engineering, Construction, Owner Operator (AECOO) sector. Web Ontology Language (OWL) (McGuinness, et al., 2004) ontology for IFC (ifcOWL) (Schevers & Drogemuller, 2005) (Beetz, et al., 2009) (Pauwels & Terkaj, 2016) is almost a literal conversion of the IFC Express schemas developed by buildingSMART and aimed to increase interoperability by sharing common data schema and exchange format.

On the other hand, ifcWoD (Djuedja & Flore, 2019) was developed to express the object-oriented constraints of the IFC schema and the semantics of the model as an OWL ontology. It is as tightly coupled with the IFC versions as ifcOWL is. Then, like ifcWoD, SimpleBIM (Pauwels & Roxin, 2016) was an attempt to create a simpler and more developer-friendly view on ifcOWL data.

There were also other ontologies. The Domotic OSGi Gateway ONTology (DOGONT) (Bonino & Corno, 2008) was initially developed to express home automation devices. In the last years, the scope has been widened to cover IoT network components.

COBieOWL (Farias, et al., 2015) is an approach to present Construction Operations Building Information Exchange (COBie) standard sheets as OWL classes and sheet columns as properties. The OWL Ontology has been automatically populated from the template using the Java-based COBieOWL tool.

BIM Shared Ontology (BIMSO) and BIM Design Ontology (BIMDO) are modular ontologies independent of the IFC standard. BIMSO is built on the UNIFORMAT II classification system, and BIMDO has its vocabulary (Niknam & Karshenas, 2017) (Rasmussen, et al., 2019).

W3C Linked Building Data Community Group (W3C LBD-CG)[1] proposed a new modular approach, where an AEC ontology, the Building Topology Ontology (BOT) (Rasmussen, et al., 2017), would not violate the W3C best practices as the previous Linked Building Data ontologies had done. For example, its design principle has been to reuse concepts in other more focused ontologies. Unlike the voluminous ifcOWL, BOT is following the best practice of keeping ontologies simple for easy maintenance.

Notable ontologies that have been used besides the BOT are Ontology for Property Management (OPM)[2], and Building Product Ontology (BPO) (Wagner & Rüppel, 2019) that allows manufacturers to define their products.

There are also domain-specific ontology suites with alignments to other LBD ontologies. Digital Construction Ontologies (DiCon) (Törmä & Zheng, 2021), developed in Diction and BIM4EEB projects, is a modularised ontology group for expressing information related to the execution of construction projects while the BIM4Ren ontologies[3] are for modelling energy-efficient renovations.

[1] https://www.w3.org/community/lbd/
[2] https://github.com/w3c-lbd-cg/opm
[3] https://models.bim4ren.eu/

2.2 Linked Building Data Converters

There have been various tools to translate BIM data models into LBD. One of the early ones was IFCtoRDF (Pauwels & Oraskari, 2016) to translate IFC STEP documents onto ifcOWL ABox format. IFCtoLBD converter is a Java-based converter developed in the Linked Building Data community. This tool extracts core BOT ontology classes and their relationships, but also, product data is present, and property values are expressed using the OPM ontology. (Rasmussen, et al., 2019) (Bonduel, et al., 2018) (Oraskari, et al., 2020).

A plug-in[4] that exports Linked Building Data from Autodesk Revit has been initially developed by Jonas Eik Bacher-Jacobsen at NIRAS and continued by Rasmussen et al. (Rasmussen, et al., 2017). It is written using the .NET developer platform. In addition to the core LBD data, the tool can export 3D spaces as OBJ encoded mesh geometry and outlines of areas as WKT polygons.

On the other hand, NIRAS IFC2BOT[5] written by Mads Rasmussen is a lightweight command-line tool written in Python 3.8. It extracts core BOT elements of an IFC document using the IfcOpenShell[6] Python library.

Moreover, in (Bourreau & Oraskari, 2021), Bourreau and Oraskari propose a novel dynamic translation method where rule-based reasoning is used for the translation.

2.3 Shapes Constraint Language (SHACL)

SHACL is a general-purpose data validation language and specification from the World Wide Web Consortium (W3C), which focuses on conformance checking of information serialised as Linked Data (Knublauch & Kontokostas, 2017a). It contains two major components: Shapes graph and Data graph. The former contains a list of user-defined constraints which are syntactically represented in SHACL language, while the latter is the RDF graph which is being validated against the Shapes graph. SHACL constraints can be used to define restrictions on the values that a property can have, the datatype of the property, numerical ranges of values, absolute string matches and also a mixture of the above. More complex constraints such as sub-graph pattern validation, conditional checking can also be expressed using SPARQL queries inside SHACL shapes, or through the use of SHACL JavaScript Functions. SHACL is a data-agnostic constraint modelling language, i.e. regardless of the underlying schema of the data, as long as the data are serialised as RDF, SHACL validation rules can be used for its conformance checks. Consequently, in the Linked Data and AEC domain, where information spans multiple domains and contains interlinks between them, SHACL is emerging as a candidate for complex semantic constraint checking.

Hagedorn and König have examined the feasibility of the rule validation approach in the AEC industry. A fictive tunnel construction project was used as a use case (Hagedorn & König, 2020), while in (Stolk & McGlinn, 2020), Stolk and McGlinn studied the possibility of using SHACL for testing ifcOWL models. Furthermore, in (Werbrouck, et al., 2019), Werbrouck et al. described how SHACL can be used like mvdXML for LBD data, while SHACL-based dynamic constraint solving has been explored for look-ahead-planning (Soman, 2019).

Prior SHACL, there was no W3C standard mechanism to check RDF data, although tools that ignored the open-world and non-unique-name assumption existed. TopBraid and Protégé are OWL aware tools. Compared with OWL, SHACL uses the closed-world assumption and separates checking data validity from inferring new facts. (Knublauch, 2017)

There have also been other approaches like plain SPARQL queries, SPARQL Inferencing Notation (SPIN), Stardoc ICD, OSLC Resource Shapes, RDF Data Descriptions, and RDFUnit to test the models. (Gayo, et al., 2017) (Gayo, et al., 2017)

When compared with ShEx[7], ShEx usually provides an enriched graph that contains a graph of valid statements, while SHACL usually focuses on validating results. The default cardinality is in SHACL zero while it is in ShEx one.

---

[4] https://github.com/MadsHolten/revit-bot-exporter
[5] https://github.com/NIRAS-MHRA/IFC2BOT
[6] http://ifcopenshell.org/python
[7] https://www.w3.org/community/shex/

3 Case: The Converter Unit Testing

When a converter program is designed and written, the process is prone to human errors caused by misinterpretations of the concepts and software bugs. Additionally, there may also be concepts that are not fully mappable.

Unit testing is a software development method to avoid unwanted errors in code when introducing new features. A collection of unit tests, where each tests a small behavioural aspect of the software, can be used to improve the code quality. When developing a program that contains complex rules to convert source data into an instance data (ABox) model that comply with the used ontologies, the output-based unit tests can be used.

To define the requirements for the unit tests, we started with the question: How to define a sound and valid conversion output?

In (Kalfoglou & Schorlemmer, 2003), Kalfoglou et al. define ontology mapping to be a way to relate two vocabularies of the same domain "in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected."

A data translator instantiates a partial ontology mapping for individuals (Euzenat, et al., 2007) (Kalfoglou & Schorlemmer, 2003). Therefore, a proper output consists of individuals with a one-to-one correspondence with an element in the input model, i.e., all individuals in the input model that have a mapping have a counterpart in the output. Accordingly, the first requirement *R1* is that the unit tests validate the ontology mapping realisation for individuals defined in the alignments.

Following Kalfoglou's definition, the relationship path and attribute sets correspondence of the inputs and output has to be tested. Correspondingly the second requirement *R2* for the unit test is to check that the translator keep the relations and the attributes of the data, i.e., is a morphism.

3.1 Requirement R1 in SHACL

There needs to be, for all input graph individuals of an ifcOWL class that is part of an alignment in the used ontologies, precisely one individual representing the same concept in the generated Linked Building Data model. In general, this subgraph matching is a known problem in theoretical computer science and is NP-complete. In the IFC context, it can be solved by comparing IFC GUIDs. Therefore the problem can be expressed: For all instances with an alignment and a GUID attribute, there has to be exactly one instance counterpart with the same GUID in the LBD graph. In SHACL-SPARQL, this can be written as shown in Listing 1.

Listing 1. SHACL rule for checking BOT class alignment

```
inst:shapeBOTClasses a sh:NodeShape ;
 sh:targetClass ifc:IfcSite, ifc:IfcBuilding,  ifc:Store;
 sh:property [
  sh:sparql [
   a sh:SPARQLConstraint ;
   sh:message "GUIDs of the aligment instances should match." ;
   sh:prefixes inst:prefixes;
   sh:select """
    SELECT $this (?guid as ?value){
     $this   ifc:globalId_IfcRoot/express:hasString ?guid .
     FILTER NOT EXISTS { ?b props:globalIdIfcRoot_attribute_simple ?guid }
     }
   """
   ]
```

```
15    ].
```

The alignment rule (Listing 1) was tested using the IFCtoLBD converter and TopBraid SHACL API[8]. In the test, the rule could validate that the aligned instances existed in the output model. The rule is specific to IFC version and OPM level (Bonduel, et al., 2018), but similar rules can be written for the other combinations.

3.2 Requirement R2 in SHACL

To validate a converter against the requirement *R2*, an approach where a generic one rule that fits all OWL alignments is not plausible. For example, owl:equivalentProperty used for property alignment has rdf:Property as both its domain and range. A similar problem is when the alignment is defined using rdfs:subPropertyOf property. Therefore they cannot be used for property paths needed for ifcOWL to BOT alignment.

Nonetheless, specific validation tests to test the existence of a known property assertion in a model can be written like shown in Listing 2.

Listing 2. SHACL rule for checking BOT relations alignment

```
1     inst:shapeBOTRelations a sh:NodeShape ;
2      sh:targetClass ifc:IfcSite;
3      sh:property [
4      sh:sparql [
5       a sh:SPARQLConstraint ;
6        sh:message "bot:hasBuilding missing." ;
7        sh:prefixes inst:prefixes;
8        sh:select """
9          SELECT $this (?guid as ?value){
10           $this   ifc:globalId_IfcRoot/express:hasString ?guid1 .
11           $this   a ifc:IfcSite .
12
13           $ifc_building   ifc:globalId_IfcRoot/express:hasString ?guid2 .
14           $ifc_building   a ifc:IfcBuilding .
15
16           ?bot_site props:globalIdIfcRoot_attribute_simple ?guid1 .
17           ?bot_building props:globalIdIfcRoot_attribute_simple ?guid2 .
18            FILTER NOT EXISTS { ?bot_site bot:hasBuilding ?bot_building }
19            }
20            """
21         ]
22      ] .
```

On the other hand, since the input models are part of the unit tests and thus unchanging, and the primary function is to track whether a change in the source code affects the output negatively, it suffices to validate that the known positive characteristics of the output model are kept. Earlier, in IFCtoRDF this has been tested comparing output with an earlier result byte-wise, which causes false-negative warnings, e.g., when Apache Jena library was updated to version 3.16.0 and white spacing was changed.

One approach to avoid oversensitive warnings and to be able to give element level error messages is to use checksums of the ordered set of triples connected to the elements or just the

[8] https://github.com/TopQuadrant/shacl

properties. This was tested in a Java unit test and the TopQuadran SHACL engine. Listing 3 shows the SHACL code. In tests, it was able to warn when there was an unknown change in the connected triple set.

Listing 3. SHACL rule to check checksum of properties

```
inst:shapeBOTChecksum a sh:NodeShape ;
  sh:targetNode inst:stairflight_ca457005-aa0d-4679-92c1-5067d702c9f3;
  sh:property [
  sh:sparql [
     a sh:SPARQLConstraint ;
     sh:message "The checksum of the properties is not valid." ;
     sh:prefixes inst:prefixes;
     sh:select """
       SELECT $this ?ResultSetHash WHERE {
       {
       SELECT $this (MD5(GROUP_CONCAT(CONCAT(STR(?p)); separator=' ')) as ?hash )
       WHERE  {
           SELECT *
           WHERE {
               $this ?p ?o.
         ORDER BY ?s ?p ?o
       } GROUP BY $this
       }
       FILTER (?hash != "82c7dc90fcb57319f2bb7ead58ead1de")
    }
    """
] .
```

Also, SHACL validation patterns can be generated programmatically from sample files. The output can be compared with one of an earlier release of the tool, or check that it is consistent with the other available LBD converters to compare the outputs. To test this, we used BOT Duplex Apartment[9] that had been exported by using Nira's Autodesk Revit LBD Plugin Exporter (Rasmussen, et al., 2017). It is an independent implementation. Both converters used the same IFC model.

The challenge was to match the individuals as the URI format differs, forming the attribute properties differently. A program to write auto-generated SHACL rules to check that the converter's graph matched in topological level was written. Also here, for this, IFC GUIDs were used to associate the individuals. Listing 4 shows a snippet of the long auto-generated SHACL rules.

Listing 4. Sample auto-generated SHACL rule

```
 inst:shape_1
a sh:NodeShape ;
sh:targetNode
 inst:building_7b7032cc-b822-417b-9aea-642906a29bd4 .
sh:property
 [ sh:hasValue
   inst:storey_7b7032cc-b822-417b-9aea-6429f95d6512 ;
  sh:minCount 1 ;
  sh:path
```

[9] https://github.com/MadsHolten/BOT-Duplex-house

```
10          bot:hasStorey
11        ] ;
12       <--- cut here --->
13       sh:property
14        [ sh:hasValue
15          inst:storey_7b7032cc-b822-417b-9aea-6429f95f770e ;
16         sh:minCount 1 ;
17         sh:path
18          bot:hasStorey
19        ] .
```

For the first requirement, *R1*, with the assumption that GUIDs are available for the elements, a general SHACL based unit test was easy to write. Requirement *R2* was more challenging than the first one. In the case of the BOT ontology with relatively few properties, the property-based approach where a rule is written for each property in the ontology is achievable. If using other LBD ontologies, the same principles apply. The drawback is that SHACL rules for numerous attribute properties would be tedious to write by hand. Hence, the presented checksum method and programmatically written SHACL rules using examples to test known inputs in a unit test are promising. Particularly, human errors caused by misinterpretations can be tracked using independent example data, preferably from the ontology designer, for test generation.

Complete source code for the test made is published in our GitHub[10] repository.

4 Case: Energy Simulation MVD use case

buildingSMART International has defined a Model View Definition (MVD) as a subset of IFC schema for a specific use case. Provided standard MVDs are, e.g. Coordination View, Structural Analysis View, Quantity Takeoff View, and Energy Analysis View (Beetz, et al., 2018).

ModelView definition written in mvdXML is specific to an IFC schema version, e.g., IFC4. It consists of ExchangeRequirements and ConceptRoots, where ExchangeRequirements define the required information to fulfil a particular task. ConceptRoots define IFC elements for which the same set of constraints apply and a collection of Concepts that assign ConceptTemplates and rules for the associated attributes. ConceptTemplates are reusable templates, a graph with the schema information embedded to describe a specific functionality, to instantiate a set of attribute values in a given IFC entity context. (Weise, et al., 2017) (Pinheiro, et al., 2018)

In SHACL, target instances can be selected using a rich set of definitions. Compared to mvdXML's applicableRootEntity and applicability rules, SHACL Advanced features sh:SPARQLTarget (Knublauch, et al., 2017b) provides similar expressivity.

Also in SHACL, reusable parts of definitions are possible. For example, external modules can be references from other SHACL definitions with owl:imports, reusable constraint components can be written using defining SHACL-SPARQL sh:ConstraintComponent constraint components, and SHACL Advanced features SHACL Triple Rules allows to infer new triples that can be used in other rules.

In an energy simulation use case, the critical aspects of an MVD view are that there should be geometry associated with the selected elements. They should have properties that are important for the energy simulation.

In our first tests using TopBraid SHACL, models could be validated to contain geometry and properties required in the exchange. Similar logical expressions as are in mvdXML rules can be written. The problem here is that SHACL is designed to focus on finding rule violation errors (Labra-Gayo, et al., 2019) which causes it to accept inputs where SHACL rule targets are missing. Therefore, the unit test needs to test the existence of the nodes separately. Data validation of ifcOWL and LBD data has been handled earlier in (Stolk & McGlinn, 2020) (Werbrouck, et al., 2019).

[10] https://github.com/jyrkioraskari/SHACL_UNITTESTS4LBD

As the model view for IFC is a valid subset of IFC schema, the model view for LBD should filter relevant data for the particular exchange scenario. Hence, checking the model is solving half of the problem. The other half should filter the data based on the exchange requirements. SHACL SPARQL Construct query and SHACL inference model can implement this. In Listing 5 is shown how to combine validation and selecting windows elements with an associated 3D geometry. Listing 6 shows a rule to list windows of a specific size.

Listing 5. Sample SHACL Filter for Window elements

```
inst:HasGeometry
    sh:path bot:hasSimple3DModel ;
    sh:minCount 1 .

inst:RuleFilterWindowsWithGeometryShape
  a sh:NodeShape ;
  sh:targetClass bot:Element ;
  sh:property inst:HasGeometry ;
  sh:rule [
    a sh:SPARQLRule;
    rdfs:label "Construct a Geometry MVD";
    sh:prefixes inst:prefixes ;
    sh:construct """
      CONSTRUCT {
        $this ?p ?o .
      }
      WHERE {
        $this ?p ?o .
        FILTER (regex (STR($this),"window_")).
      }
      """
  ] .
```

Listing 6. Sample SHACL Filter for Window elements

```
inst:RuleFilterShape
 a sh:NodeShape ;
 sh:targetClass beo:Window ;
 sh:rule [
   a sh:SPARQLRule;
   rdfs:label "Constructs an MVD";
   sh:prefixes inst:prefixes ;
   sh:construct """
      CONSTRUCT {
        $this ?p ?o .
      }
      WHERE {
       $this props:objectTypeIfcObject_attribute_simple  "2800mm x 2410mm" .
       $this ?p ?o .
      }
      """
] .
```

SHACL SPARQL Construct query can be used to develop MVD view, but it has also limitations. While the purpose is to create a filtered list of validated triples, re-creating new triples is "counterproductive" since always, when creating something, the result should be validated. Also, the SPARQL Construct query is best suited for simple copies. A query to copy triples recursively for a validated building element is not elementary. It is recommended to use OPM level 1 (Bonduel, et al., 2018) and avoid blank nodes. Future studies will study how this can be developed further and focus on the best granularity and algorithm of the graph split separately. The principles used here are generic and well suitable for any LBD ontology.

5 Conclusion and Future work

In this work, we presented an exploration of unit tests that can be framed and carried out to check the conformance of Linked Building Data using SHACL. As mentioned previously, this work is part of the EU-funded BIM4Ren project. Different tools are leveraged to form workflows that can be used to improve processes in the renovation phase. Conformance tests such as those presented in this paper play an essential role in checking that the RDF data contains the necessary information for tools that would use them. Since such conformance tests can be used in any tasks that require checking RDF data, the methodology and findings from this work can be extrapolated and used for any tool that outputs RDF data.

Besides the MVD checking and filtering of the Linked Building Data model, unit tests in a converter have a role in ensuring no data loss or corruption in the translation process. So far, there have not been unit tests that focus on the ontology translation process of an LBD model. This study presents a couple of unit test methods to improve conversion quality. In this feasibility study, we have shown that unit tests can be written for the key aspects of the conversion. After this feasibility study, a more extensive analysis will be made.

In the last decade, one of the big problems in LBD domain has been to manage the vast amount of building-related data and the difficulty to separate the wheat from the chaff. This work shows that besides validation, model views can be created for LBD data using SHACL. Although the used SHACL SPARQL Construct method works, it also has limitations. Re-creating the data add a new process layer and breaks against the idea that the tool is only filtering the content, but It also opens new possibilities for further conversions and to simplify the data for the view. In future studies, we show how standard model views can be created using SHACL.

Acknowledgements

This research is part of the BIM4Ren project which has received funding from the European Union's Horizon H2020 research and innovation programme under grant agreement No 820773.

References

Beetz, J., Borrmann, A. & Weise, M., 2018. Process-based definition of model content. In: Building Information Modeling. s.l.:Springer, pp. 127-138.

Beetz, J., Leeuwen, J. & Vries, B., 2009. IfcOWL: A case of transforming EXPRESS schemas into ontologies. AI EDAM, Volume 23, pp. 89-101.

Bonduel, M. et al., 2018. The IFC to linked building data converter: current status. s.l., s.n., pp. 34-43.

Bonino, D. & Corno, F., 2008. Dogont-ontology modeling for intelligent domotic environments. s.l., s.n., pp. 790-803.

Bourreau, P. & Oraskari, J., 2021. BIM format conversion as inference-based ontology alignment. Manuscript submitted for publication. LDAC 2021, Luxenburg.

Djuedja, T. & Flore, J., 2019. Information modelling for the development of sustainable construction (MINDOC), s.l.: s.n.

Euzenat, J., Shvaiko, P. & others, 2007. Ontology matching. s.l.:Springer.

Farias, M. T., Roxin, A. & Nicolle, C., 2015. Cobieowl, an owl ontology based on cobie standard.

s.l., s.n., pp. 361-377.
Hagedorn, P. & König, M., 2020. Rule-Based Semantic Validation for Standardized Linked Building Models. s.l., s.n., pp. 772-787.
Kalfoglou, Y. & Schorlemmer, M., 2003. Ontology mapping: the state of the art. The Knowledge Engineering Review, Volume 18, p. 1.
Knublauch, H., Allemang, D. & Steyskal, S., 2017b. SHACL Advanced Features. [Online] Available at: https://www.w3.org/TR/shacl-af/ [Accessed 6 5 2021].
Knublauch, H. & Kontokostas, D., 2017a. Shapes Constraint Language (SHACL). [Online] Available at: https://www.w3.org/TR/shacl/[Accessed 6 5 2021].
Labra-Gayo, J. E., García-González, H., Fernández-Alvarez, D. & Prud'hommeaux, E., 2019. Challenges in RDF validation. In: Current Trends in Semantic Web Technologies: Theory and Practice. s.l.:Springer, pp. 121-151.
McGuinness, D. L., Van Harmelen, F. & others, 2004. OWL web ontology language overview. W3C recommendation, Volume 10, p. 2004.
Niknam, M. & Karshenas, S., 2017. A shared ontology approach to semantic representation of BIM data. Automation in Construction, Volume 80, pp. 22-36.
Oraskari, J. et al., 2020. jyrkioraskari/IFCtoLBD: IFCtoLBD 2.5. [Online] Available at: https://doi.org/10.5281/zenodo.4056940
Pauwels, P., Krijnen, T., Terkaj, W. & Beetz, J., 2017. Enhancing the ifcOWL ontology with an alternative representation for geometric data. Automation in Construction, Volume 80, pp. 77-94.
Pauwels, P. & Oraskari, J., 2016. IFC-to-RDF Converter. URL: https://github. com/IDLabResearch/IFC-to-RDF-converter (Letzter Zugriff am: 24.03. 2016).
Pauwels, P. & Roxin, A., 2016. SimpleBIM: From full ifcOWL graphs to simplified building graphs. s.l., s.n., pp. 11-18.
Pauwels, P. & Terkaj, W., 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction, Volume 63, pp. 100-133.
Pinheiro, S. et al., 2018. MVD based information exchange between BIM and building energy performance simulation. Automation in Construction, Volume 90, pp. 91-103.
Rasmussen, M. H., Hviid, C. A. & Karlshøj, J., 2017. Web-based topology queries on a BIM model.
Rasmussen, M. H. et al., 2019. Managing interrelated project information in AEC Knowledge Graphs. Automation in Construction, Volume 108, p. 102956.
Rasmussen, M. H., Lefrançois, M., Schneider, G. F. & Pauwels, P., 2019. BOT: the building topology ontology of the W3C linked building data group. Semantic Web, pp. 1-19.
Rasmussen, M. H., Pauwels, P., Hviid, C. A. & Karlshoj, J., 2017. Proposing a central AEC ontology that allows for domain specific extensions. s.l., s.n., pp. 237-244.
Schevers, H. & Drogemuller, R., 2005. Converting the industry foundation classes to the web ontology language. s.l., s.n., pp. 73-73.
Soman, R. K., 2019. Linked-Data Based Dynamic Constraint Solving Framework to Support Look-Ahead-Planning in Construction. Newcastle-upon-Tyne, s.n., pp. 871-880.
Stolk, S. & McGlinn, K., 2020. Validation of IfcOWL datasets using SHACL.
Törmä, S. & Zheng, Y., 2021. Digital Construction Ontologies (DiCon). s.l.:s.n.
Wagner, A. & Rüppel, U., 2019. BPO: the building product ontology for assembled products. s.l., s.n., p. 12.
Weise, M., Liebich, T., Nisbet, N. & Benghi, C., 2017. IFC model checking based on mvdXML 1.1. eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2016, pp. 19-26.
Werbrouck, J., Senthilvel, M., Beetz, J. & Pauwels, P., 2019. A checking approach for distributed building data. s.l., s.n., p. 173.