Querying and reasoning over large scale building data sets: an outline of a performance benchmark

Pieter Pauwels
Department of Architecture and Urban Planning
Ghent University
J. Plateaustraat 22
B-9000 Ghent, Belgium
pipauwel.pauwels@ugent.be

Tarcisio Mendes de Farias
Active3D
Dijon, France
tarcisio.mendesde-farias@checksem.fr

Chi Zhang
Department Built Environment
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB Eindhoven,
The Netherlands
c.zhang@tue.nl

Ana Roxin
Checksem, Laboratory LE2I (UMR CNRS 6306)
University of Burgundy
Dijon, France
ana-maria.roxin@u-bourgogne.fr

Jakob Beetz
Department Built Environment
Eindhoven University of Technology
P.O. Box 513
NL-5600 MB Eindhoven,
The Netherlands
j.beetz@bwk.tue.nl

Jos De Roo
Agfa HealthCare NV
Moutstraat 100
B-9000 Ghent, Belgium
jos.deroo@agfa.com

ABSTRACT

The architectural design and construction domains work on a daily basis with massive amounts of data. Properly managing, exchanging and exploiting these data is an ever ongoing challenge in this domain. This has resulted in large semantic RDF graphs that are to be combined with a significant number of other data sets (building product catalogues, regulation data, geometric point cloud data, simulation data, sensor data), thus making an already huge dataset even larger. Making these big data available at high performance rates and speeds and into the correct (intuitive) formats is therefore an incredibly high challenge in this domain. Yet, hardly any benchmark is available for this industry that (1) gives an overview of the kind of data typically handled in this domain; and (2) that lists the query and reasoning performance results in handling these data. In this article, we therefore present a set of available sample data that explicates the scale of the situation, and we additionally perform a query and reasoning performance benchmark. This results not only in an initial set of quantitative performance results, but also in recommendations in implementing a web-based system relying heavily on large semantic data. As such, we propose an initial benchmark through which new upcoming data management proposals in the architectural design and construction domains can be measured.

SBD'16, July 01 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2928294.2928303

CCS Concepts

•**Computing methodologies** → **Knowledge representation and reasoning;** •**Information systems** → **Semantic web description languages;** *Information retrieval query processing;* •**Theory of computation** → *Semantics and reasoning;* •**Applied computing** → Computer-aided design;

Keywords

big data; benchmark; reasoning; semantic web; building information modeling; built environment; IFC; OWL

1. INTRODUCTION

With the advent of Building Information Modelling (BIM) tools [14], fundamentally new processes evolve in order to allow large amounts of building information to be managed. Building information management occurs in many forms. For example, information needs to be modeled comprehensively and visually by the architect; information needs to be handed over from architect to engineering office and to contractors and subcontractors; information modeled by a subcontractor is used for automated energy performance checking and other kinds of building performance checking (acoustics, structural analysis, cost calculation, planning, ...); and so forth. In all of these cases, the building information needs to be made available in another form (language, syntax, semantics, level of detail).

To make this possible, many of the BIM-related initiatives rely on a neutral, interoperable representation of the BIM model, namely the Industry Foundation Classes (IFC) standard [17], which is developed and maintained by the international buildingSMART organization. However, it is far from straightforward to map from this neutral representation, which is defined using the EXPRESS information modeling language [15], to the diverse data models used in contexts as diverse as the examples outlined above. It has

been proposed that such a mapping task is easier to implement by relying on semantic web techniques [22, 10, 12].

The IFC standard is now also available as a Web Ontology Language (OWL) ontology (ifcOWL [20, 7]), joining existing ontologies for the built environment. As a result, building information is more commonly made available using RDF and OWL. One example application aims at accommodating acoustic regulation compliance checking for BIM models [23]. An indication is made of the way in which rules (RBox) can be represented and used in combination with a domain ontology (TBox) and an instance model (ABox), so that an inference engine can immediately indicate whether a building model is compliant or not with the European acoustic regulations. In the area of Health and Safety (HS) measures, the Job Hazard Analysis (JHA) application proposed by [25] provides another use case. The authors propose to combine an RDF representation of the building model with a number of ontologies and SWRL rules that allow analyzing the construction project in terms of jobs, tasks, safety procedures, and resources required to allow the safe execution of these job steps. Many similar examples are available.

A number of these examples rely on semantic data enrichment or schema and data transformations. Namely, the information needs to be used in a syntax and semantics that is different from the one provided (typically IFC). Data transformation processes are for example proposed in [23] and [4] to transform IFC data into regulatory data models. These kinds of transformation are to some extent similar to what was proposed in [22] regarding the automatic transformation of IFC data to X3D data and to STL data (and back). Furthermore, the usage of SWRL rules was also proposed in combination with the ifcOWL ontology [10, 12], thus allowing to automatically transform data patterns (subset graphs) into parallel and in this case less complex data patterns. In this last case, it is explicitly proposed to transform the building data at query time, using SWRL rules that are triggered on request.

At the core of the above approaches are three key components: (1) a schema (OWL ontology) that defines what kind of information is used by the rule checking process and how it is structured (the TBox), (2) a set of instances (RDF graphs) asserting facts based on the concepts defined in the TBox (the ABox), and (3) a set of rules (e.g. `IF-THEN` statements) that can be directly combined with the schema (the RBox). Declarative data transformation procedures are then accessible as soon as all the data and all the rules are available in a complete and consistent shape: inferences are generated by generic reasoning engines, the results are asserted as new facts into the graph, after which they are used in specific applications (e.g. simple visualisation in a graphical user interface; job hazard analysis; acoustic building performance checking). Depending on the rules that are being triggered, one set of information then has the potential to be made available in a diverse number of forms (see [11]), bringing an entirely new form of interoperability for an industry that has always relied heavily on the combination of an agreed standard with many intransparent import and export procedures that were implemented using procedural programming languages. Namely, with its logical basis in Description Logics (DL) [2], a logic-based semantic big data publishing approach emerges.

The three components listed above (RBox, TBox, ABox) can be realized in various ways, however, depending on the approach taken and the software systems used. Diverse reasoning engines are available, diverse query processing techniques are available, queries can be handled at various moments in time, the size as well as the complexity of the data varies, and so forth. Any of these choices typically impacts on expressiveness versus performance. In order to make informed choices in real-world use case scenario's, there is a high need for an appropriate rule and query execution *performance benchmark* in the direct context of the data, ontology, rules and queries used in this particular industry, which is precisely the aim of this paper. A solid benchmark or reference point is aimed at, illustrating at least an initial outline of (1) the kind of data that is typically available in this industry, (2) as well as an initial assessment of the performance results for accessing these large scale building data. Although we have to limit here to an overview position statement, we do aim to further extend this study, to make full data available and provide in-depth documentation of the results.

2. CONSTRUCTING A PERFORMANCE BENCHMARK

Although various semantic applications have been proposed in the architectural design and construction industry, no indications are typically given about the performance of the system. We aim to remedy this situation with the performance benchmark proposed in the remainder of this article. We will hereby consider three approaches that allow to transform data (RBox rules) and make them available (queries), namely SPIN and Jena [1], EYE [13] and Stardog [9]. In order to properly evaluate the three semantic data access procedures, we have set up a test environment (TBox, RBox, ABox) that allows to compare the test results in a quantitative manner. This test environment consists of a TBox, namely the ifcOWL ontology (Sect. 2.1); an ABox, namely a set of 369 ifcOWL-compliant building models (Sect. 2.2); an RBox, namely a set of 68 data transformation rules (Sect. 2.3); and a set of simple queries (Sect. 2.4). All public data for this test is made available online [19].

2.1 Presentation of the ifcOWL ontology

In our test environment, all building models are encoded using the ifcOWL ontology. This ontology has been built up under the impulse of numerous initiatives during the last ten years [24, 5, 3]. The most complete overview of the key decisions in constructing this ifcOWL ontology is documented in [20]. The ontology that was used for this test is the one that is made publicly available by the buildingSMART Linked Data Working Group (LDWG) [8].

2.2 Building models used for testing

The test environment has one common set of building models following the ifcOWL ontology. These are building information models that were modelled by people outside the current research team. They have been modelled in different BIM modelling environments, including most prominently Tekla Structures (Trimble) and Autodesk Revit, which are both BIM authoring tools commonly used in construction industry. The models were exported to IFC and made available for this project. A number of BIM models are publicly available [18], whereas other models are not disclosed (private models). The test models can be categorised

in a number of dimensions: BIM authoring tool, model size and IP level. In total, our test set includes 5 undisclosed IFC models and 364 publicly available models (IP level). The 5 restricted IFC models were modelled using Autodesk Revit 2012. The publicly available IFC models were modelled using a range of different BIM authoring tools. Table 1 gives an overview of how many public files were available for each BIM environment found (retrieved from IFC file metadata). Most public IFC files were modelled using Tekla Structures (227 out of 369 - 61.5%), followed by unknown or manual (38 out of 369 - 10.3%) and Autodesk Revit (27 out of 369 - 7.3%).

BIM environment	# files
Tekla Structures	227
unknown or manual	38
Autodesk Revit	27
Xella BIM	15
Autodesk AutoCAD	12
iTConcrete	9
SDS	8
Nemetschek AllPlan	7
GraphiSoft ArchiCAD	5
Various others	21

Table 1: Number of public files available for each software environment.

The test files are also quite different in terms of model size. In this regard, we have set an arbitrary distinction between small models (0 to 500,000 IFC instances), medium models (500,000 to 2 million IFC instances), and large models (more than 2 million IFC instances). Most models are relatively small, but a number of larger models were tested as well in this performance benchmark. Table 2 gives an overview of the number of IFC files for each model size range, private as well as public. An indication is also given of the average associated file size (in MB). Note that the 321 small-sized IFC models follow an exponential curve. In fact, 222 IFC models have less than 30,000 IFC instances (av. equivalent of 1.5MB). In total, the test set contains 85,209,175 IFC instances.

# IFC instances	av. file size	# files
0 - 500,000	0 - 30MB	321
500,000 - 2,000,000	30 - 100MB	37
> 2,000,000	> 100MB	11

Table 2: Number of files for each model size (small, medium, large).

Each building model was originally available in IFC2X3. These building models were converted to ifcOWL-compliant RDF graphs using the software made available at [18]. This conversion is done prior to the actual performance benchmark test. The conversion resulted in an RDF graph (TTL syntax) for each of the IFC files considered. The 85,209,175 IFC instances result in at least as many RDF instances in the ABox.

2.3 Rules used for testing

The performance benchmark experiment relies not only on a representative set of building models and ontology, but also on a representative set of rewrite rules. For this experiment, we have therefore manually built a set of 68 rewrite rules. These 68 rules can be classified in a number of types, depending on their content, as illustrated by Table 3. All rules are inspired by the work on simplification of ifcOWL graphs, as suggested and initially documented in [10, 12, 21, 6].

Rule set	Description
RS1	Contains 2 rules for rewriting property set references into additional property statements `sbd:hasPropertySet` and `sbd:hasProperty`. This is a small, yet often used rule set.
RS2	Includes 31 rules, all involving subtypes of the `IfcRelationship` class.
RS3	Contains 3 rules related to handling lists in IFC.
RS4	Contains one rule that allows wrapping simple data types.
RS5	Consists of 20 rules for inferring single property statements `sbd:hasPropertySet` and `sbd:hasProperty`.
RS6	Extends Rule Set 5 and Rule Set 1 with 6 additional rules for inferring whether an objet is internal or external to a building.
RS7	Contains 7 rules dealing with the (de)composition of building spaces and spatial elements.

Table 3: Description of the Rule Sets considered for testing.

2.4 Queries used for testing

We have built a limited list of 60 queries, each of which is able to trigger at least one of the available rules. As the focus of this article is primarily on execution performance in combination with data transformation, the considered queries are entirely based on the right-hand sides of the rules in Sect. 2.3. For this article, we limit ourselves to the 3 queries that are listed in Table 4: a simple query with little results (Q1), a simple query with many results (Q2), and a complex query that triggers a considerable number of rules (Q3).

Query	Query contents
Q1	?obj sbd:hasProperty ?p
Q2	?point sbd:hasCoordinateX ?x . ?point sbd:hasCoordinateY ?y . ?point sbd:hasCoordinateZ ?z
Q3	?d rdf:type sbd:ExternalWall

Table 4: List of queries considered for testing.

2.5 Test environment

Each of the queries in Table 4 is run over each of the available building models, while taking into account the ontology and the available rules. This occurs in one central server that serves as the test environment. This server was supplied by the University of Burgundy, research group CheckSem, and had the following specifications: Ubuntu OS, Intel Xeon CPU E5-2430 at 2.2GHz, 6 cores and 16GB of DDR3 RAM memory. Three Virtual Machines (VMs) were set up in this central server and managed as separate test environments. Each of these VMs had 2 cores out of 6 allocated and each contained the above resources (ontologies, data, rules, queries). A first VM (SPIN VM) implemented Jena TDB [1], a second VM (EYE VM) used EYE inference engine [13], and finally a third VM (STARDOG VM) used a Stardog triplestore [9].

2.5.1 SPIN VM

The SPIN VM is implemented based on the open source APIs of Topbraid SPIN (SPIN API 1.4.0) and Apache Jena (Jena Core 2.11.0, Jena ARQ 2.11.0, Jena TDB 1.0.0) [16, 1]. Rules listed in Sect. 2.3 are written in SPARQL with Topbraid Composer Free version, and they are exported as RDF Turtle files used in the test environment. A small Java program is implemented to read RDF models, schema, rules from the TDB store and query data. All the SPARQL queries are configured using the jena.sparql.algebra package to optimize the processing sequence of their triple patterns in order to achieve a better performance.

Rules are written in two ways using the SPIN framework. Forward SPIN rules are firstly preprocessed by the SPIN engine. These rules which assert SPIN functions or rules in OWL ontologies will fire them using the SPIN engine and the Jena engine respectively. This process continues recursively until no SPIN function is called. Generated triples will go through this process iteratively until no new facts are generated in one round. In the query run time, the SPARQL query instances also fire SPIN functions or OWL rules if they are related to the query.

To avoid unnecessary reasoning processes, in this test environment the RDF instances and ifcOWL are combined as an object of org.apache.jena.ontology.OntModel with the specification of OntModelSpec.RDFS MEM RDFS INF. It means that, although the ifcOWL ontology has used the OWL DL profile, only the RDFS vocabulary is supported.

2.5.2 EYE VM

In this test, we used 'EYE-Winter16.0302.1557', which relies on 'SWI-Prolog 7.2.3 (amd64): Aug 25 2015, 12:24:59'. EYE is a semibackward reasoner enhanced with Euler path detection [13]. Semi-backward reasoning is backward reasoning for EYE components, i.e. rules using <= in N3, and forward reasoning for rules using => in N3. As our rule set currently contains only rules using =>, forward reasoning will take place. Except for the building models (data) and ifcOWL ontology (IFC2X3_TC1.ttl), only N3 statements were used. Individual commands have been fired towards the reasoning engine, thus mimicking user interaction (cfr. API calls, query statements). Each command is executed five times to allow calculating the average query execution time. Each command includes the full ontology, the full set of rules and the RDFS vocabulary, as well as one of the 369 building model files and one of the 3 query files. In this approach, no triple store is used: triples are processed directly from the considered files.

2.5.3 Stardog VM

This approach is implemented using a 4.0.2 Stardog semantic graph database [9]. Stardog implements an OWL reasoner associated to a rule engine. Stardog supports SWRL rules, allows backward-chaining reasoning and is a large-scale triple store. Using backward-chaining reasoning allows avoiding triple materialization, thus saving query execution time. Stardog 4.0.2 is implemented in Java 8, and supports the RDF 1.1 graph data model, OWL2 profiles and SPARQL 1.1. Stardog allows using user-defined rules for inference, along with closed-world reasoning capabilities. Stardog performs reasoning by applying a query rewriting approach. In this approach, SWRL rules are taken into account during the query rewriting process.

3. RESULTS AND CONCLUSIONS

Each of the three procedures tested in this paper has its own specifics and characteristics. As all three procedures rely on components that are implemented in very diverse ways (commercial application versus Prolog implementation versus an approach relying on the Jena software libraries), the performance results for the three procedures are not directly comparable. Although we do present indicative query execution times, the creation of a full benchmark will require further work. The main focus of the work presented here is to give an initial indication of the performance differences that can be obtained for three example implementation approaches. Our conclusion outlines an unprecedented list of key decisions and choices for anyone willing to implement a semantic rule-checking process based on semantic web technologies in construction industry.

It is impossible to outline results for all 369 building models, 68 rules, and 60 queries. Hence, we limit ourselves to 6 hand-picked building models of varying size for listing the average query execution times (AQT) for each of the three procedures (Table 5). In addition, the graphs in Fig. 1 and 2 plot the query execution times in this Table with the corresponding result counts, for Q1 and Q2 respectively. Note that the table only lists query execution times. In the SPIN approach, this means that for Q1 and Q2, the execution times are given for a backward-chaining inference process plus actual query execution time; whereas for Q3, the execution times are given for the query execution time itself, as the forward-chaining inference process takes place beforehand. In the EYE approach, this means that time required for network traffic, which can be compared to the time required to load the resources into a triple store, is not taken into account. In the Stardog approach, the displayed execution times include the backward-chaining inference process as well as the actual query execution time.

query	model	SPIN (s)	EYE (s)	Stardog (s)
Q1	Model1	135.36	37.11	13.44
Q1	Model2	1.47	0.29	0.17
Q1	Model3	24.01	4.87	1.4
Q1	Model4	41.28	12.95	3.55
Q1	Model5	4.99	1.05	0.33
Q1	Model6	0.55	0.16	0.08
Q2	Model1	46.17	2.10	6.82
Q2	Model2	92.03	4.20	15.83
Q2	Model3	82.68	4.12	15.28
Q2	Model4	19.93	1.04	2.81
Q2	Model5	3.69	0.21	1.36
Q2	Model6	0.74	0.045	1.00
Q3	Model1	0.001	0.001	0.07
Q3	Model2	0.006	0.003	0.12
Q3	Model3	0.002	0.003	0.31
Q3	Model4	0.005	0.001	0.20
Q3	Model5	0.006	0.013	0.20
Q3	Model6	0.001	0.001	0.13

Table 5: Query performance results for 3 queries, 6 models, and 3 systems.

A considerable number of findings can be made from this test. Many of these findings return in each of the three considered approaches. Hence, we summarise them here, in order of impact on performance, to indicate key aspects in implementing a system that provides access to large semantic data sets, while also allowing to transform the data on demand using rules.

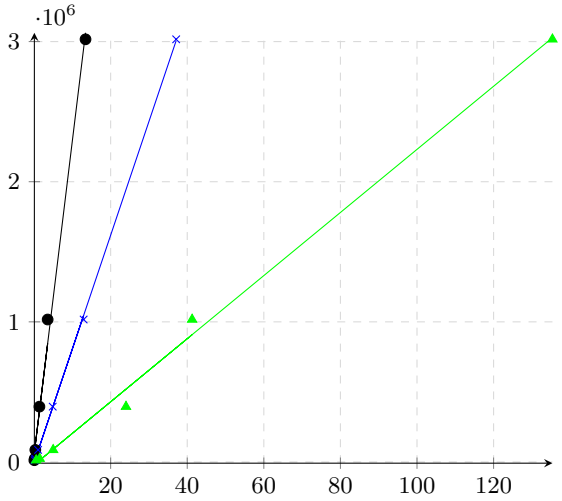

Figure 1: Plot showing the linear relation between query time (x-axis) and result count (y-axis) for query Q1 in each of the three implementation procedures (green = SPIN; blue = EYE; black = Stardog).

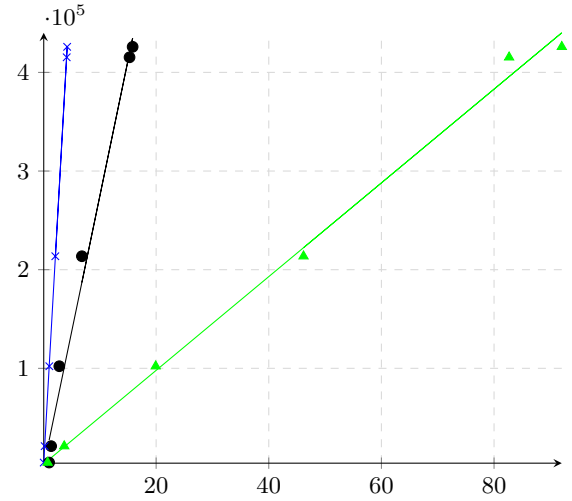

Figure 2: Plot showing the linear relation between query time (x-axis) and result count (y-axis) for query Q2 in each of the three implementation procedures (green = SPIN; blue = EYE; black = Stardog).

1. *Indexing algorithms, query rewriting techniques, and rule handling strategies*
   The overall strategy in which rules, queries and data are actually handled by the software systems (triple store, inference engine, software components) obviously has the highest impact on performance. In this regard, the three considered procedures are quite far apart from each other, explaining the considerable performance differences, not only between the procedures (Table 5), but also between diverse usages within one and the same system. For each of the three approaches, it is not entirely known which algorithms and optimisation techniques are used, apart from the limited amounts of information communicated in the system documentation files. As a result, it is not really possible to make a fair comparison between the considered approaches concerning differences in indexation algorithms, query rewriting techniques and rule handling strategies used.

2. *Forward-chaining versus backward-chaining*
   The difference between a forward-chaining and a backward-chaining approach is very important. The impact in making this choice is most notably seen in the first column in Table 5, which shows the same procedure (SPIN) both in backward-chaining (Q1 and Q2) and forward-chaining (Q3) mode. For Q1 and Q2, the inference engine thus still needs to process the rules at query execution time. However, as it only needs to process those rules that are relevant considering the provided query, this can occur relatively fast. Depending on the engine (see first item above), speed can be further increased, as is shown in the performance results for Q1 and Q2 by Stardog (column 3 in Table 5). These queries trigger backward-chaining reasoning processes in Stardog as well, but they finish more rapidly. The disadvantage of forward-chaining reasoning process is that millions of triples could be materialised, which is the case for the tested building models in the EYE VM.

3. *The dependency on the kind of data available in the models*
   A relation exists as well between query execution time and the kind of data that is queried for. This can be found in the performance differences between Q2 and Q3. Query Q3 triggers a rule that in turn triggers several other rules in the rule set. If the first rule does not fire, however, the process stops early. Query Q2, however, fires relatively long rules. It takes more time to make these matches in all three approaches.

4. *The effect of using a triple store*
   The usage of a triple store considerably increases query execution time performance. Loading files in memory at query execution time leads to considerable delays. This is not displayed in Table 5, as this table only takes into account the actual query execution times, not loading times.

5. *The dependency on the number of output results*
   All three approaches indicated that there is a direct relation between query execution time and the number of results that is eventually retrieved (see also Fig. 3 and 3. Indeed, as more results are available, more triples need to be matched, leading to more assertions. The relation between query execution time and the number of results is linear.

Future work consists of further elaborating this initial performance benchmark with additional data and rules and comparing results on a wider scale for the individual approaches separately, as well as with other approaches not considered here.

4. ACKNOWLEDGMENTS

The authors would like to acknowledge the Special Research Fund (BOF) of Ghent University, the China Scholarship Council (CSC), the Burgundy Regional Council, and the French company ACTIVe3D (http://www.active3d.net/).

5. ADDITIONAL AUTHORS

Christophe Nicolle, Checksem, Laboratory LE2I (UMR CNRS 6306), University of Burgundy, email: `cnicolle@u-bourgogne.fr`.

6. REFERENCES

[1] Apache. Jena, 2015. https://jena.apache.org/.

[2] F. Baader and W. Nutt. Basic description logics. In *Description Logic Handbook: Theory, Implementation, and Applications*, pages 47–100. Cambridge University Press, Cambridge, MA, USA, 2003.

[3] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou, and R. D. Sriram. OntoSTEP: Enriching product model data using ontologies. *Computer-Aided Design*, 44(6):575–590, 2012.

[4] T. H. Beach, Y. Rezgui, H. Li, and T. Kasim. A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert Systems with Applications*, 42:5219–5231, 2015. http://dx.doi.org/10.1016/j.eswa.2015.02.029.

[5] J. Beetz, J. Van Leeuwen, and B. de Vries. IfcOWL: a case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(1):89–101, 2009.

[6] S. Borgo, E. M. Sanfilippo, A. Sojic, and W. Terkaj. Ontological analysis and engineering standards: an initial study of IFC. In *Ontology Modeling in Physical Asset Integrity Managemen*, pages 17–43. Springer, 2015.

[7] BuildingSMART International. Overview page for the linked data working group. http://www.buildingsmart-tech.org/future/linked-data/.

[8] BuildingSMART International. IFC2X3_TC1 - OWL ontology for the IFC conceptual data schema and exchange file format for Building Information Model (BIM) data, 2015. http://www.buildingsmart-tech.org/future/linked-data/ifcowl/20150925_latest/IFC2X3_TC1.owl.

[9] Complexible Inc. Stardog 4: The manual. http://docs.stardog.com/.

[10] T. M. de Farias, A. Roxin, and C. Nicolle. A rule based system for semantical enrichment of building information exchange. In *CEUR Proceedings of RuleML (4th Doctoral Consortium)*, volume 1211, pages 2–9, 2014.

[11] T. M. de Farias, A. Roxin, and C. Nicolle. FOWLA, a federated architecture for ontologies. In *RuleML 2015*, volume 9202 of *Lecture Notes in Computer Science (LNCS)*, pages 97–111. Springer, 2015.

[12] T. M. de Farias, A. Roxin, and C. Nicolle. IfcWoD, semantically adapting IFC model relations into OWL properties. In *Proceedings of the 32rd International CIB W78 Conference*, pages 175–185, Eindhoven, NL, 2015.

[13] J. De Roo. Euler Yet another proof Engine. http://eulersharp.sourceforge.net/.

[14] C. M. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM handbook: a guide to building information modeling for owners, managers, architects, engineers, contractors, and fabricators*. John Wiley & Sons, Hoboken, NJ, USA, 2008.

[15] International Organization for Standardization. ISO 10303-11: Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual, 2004.

[16] H. Knublauch, J. A. Hendler, and K. Idehen. SPIN - Overview and Motivation - W3C Member Submission 22 February 2011. http://www.w3.org/Submission/spin-overview/.

[17] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, S. Richter, T. Chipman, M. Weise, and J. Wix. Industry Foundation Classes IFC4 official release, 2013. http://www.buildingsmart-tech.org/ifc/IFC4/final/html/index.htm.

[18] P. Pauwels. IFC repository. http://smartlab1.elis.ugent.be:8889/IFC-repo/.

[19] P. Pauwels, T. Mendes de Farias, C. Zhang, A. Roxin, J. Beetz, J. De Roo, and C. Nicolle. Semantic Big Data (SBD) Workshop at ACM SIGMOD 2016 - additional data. http://users.ugent.be/~pipauwel/SBD2016_PerfBench/SBD2016_additionaldata.html.

[20] P. Pauwels and W. Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

[21] P. Pauwels, W. Terkaj, T. Krijnen, and J. Beetz. Coping with lists in the ifcowl ontology. In *Proceedings of the 22nd EG-ICE International Workshop*, pages 113–122, Eindhoven, Netherlands, 2015.

[22] P. Pauwels, D. Van Deursen, J. De Roo, T. Van Ackere, R. De Meyer, R. Van de Walle, and J. Van Campenhout. Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 25(4):317–332, 2011.

[23] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, and J. Van Campenhout. A semantic rule checking environment for building performance checking. *Automation in Construction*, 20(5):506–518, 2011.

[24] H. Schevers and R. Drogemuller. Converting the Industry Foundation Classes to the Web Ontology Language. In *Proceedings of the First International Conference on Semantics, Knowledge and Grid*, pages 556–560, Washington, DC, 2005. IEEE Computer Society.

[25] S. Zhang, F. Boukamp, and J. Teizer. Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA). *Automation in Construction*, 52:29–41, 2015.