Semantic BIM Reasoner for the verification of IFC Models

M. Fahad, Bruno Fies, Nicolas Bus

Paper published in: Proceedings of the 12th European conference on product and process modelling (ECPPM 2018), Copenhagen, Denmark, September 12-14, 2018, eWork and eBusiness in Architecture, Engineering and Construction - Jan Karlshoj, Raimar Scherer (Eds) Taylor & Francis Group, [ISBN 9781138584136]

Semantic BIM Reasoner for the verification of IFC Models

M. Fahad
Experis IT, 1240 Route des Dolines, 06560 Valbonne, France

N. Bus & B. Fies
CSTB - Centre Scientifique et Technique du Bâtiment, 290 Route de Lucioles, 06560 Valbonne, France

ABSTRACT: Recent years have witnessed the development of various techniques and tools for the building code-compliance of IFC models. Indeed these are great efforts, but, still there is a gap for the fully automatic building code-compliance. This paper presents our research and development of *Semantic BIM Reasoner (SBIM-Reasoner)* which employs semantic technologies to meet the requirements of semantic verification of an IFC model. *SBIM-Reasoner* employs several preprocessors (IFC to RDF converter, Geometry Extractor) to build the semantic repository from the input IFC model. Once all the triples are generated from the initial data (.ifc file), Stardog is used to build a knowledge graph for the semantic verification. All types of inference and reasoning mechanisms for the semantic verification are applied over this knowledge graph to meet the requirements of verification. Knowledge graph over triplets enables freedom of extending RDF based Semantic IFC model, creation of newer vocabulary and formation of newer rules, concatenation of triplets to build rules with condition and constraints over IFC data, dynamic reasoning over the triplets based on the initial data of IFC model, etc. Finally, we tested our prototype by using several online IFC models. We conclude that semantic technologies provide more rich mechanisms and answer vast types of queries for the verification of IFC models. It provides powerful features based on SPARQL libraries and serves best for the automated code compliance and verification of IFC models.

1 INTRODUCTION

Building Information Modeling[1] (BIM) is to understand a building through the usage of a digital model which draws on a range of data assembled collaboratively before, during and after construction [1]. BIM with its interoperability properties is intended to facilitate exchanges and handovers between different stakeholders. Whereas the visualization and geometric representation are intrinsic to the digital building model, the fields of quality requirements, evaluation and regulatory contextualization (destination, named areas, threshold values, certified data, evidence of compliance, etc.) need higher level of maturity [2]. *Industry Foundation Classes* (IFC), based on a neutral format, is a complete and fully stable open and international standard for exchanging BIM data [3]. *Code Compliance* checking of BIM is necessary in order to provide stake-holders a high quality *IFC* model that ensures accurate, consistent and reliable results in the entire life-cycle of BIM. Verification of IFC models for the code compliance checking is one of the hot challenges of the present decade. Different approaches and tools are already contributed for the automated code compliance checking [4].

Our enterprise, *CSTB*, through its research aims at automating *French Building Code Compliance* as much as possible, or at least improves the control of regulations from a digital model design phase. Its goal is to provide automatic requirements verification to warn the non-conformities with the associated 3D visualization, or to provide access to the technical documentation for a given digital model based on its sophisticated contextual information. This paper presents our several contributions towards this research. First, we analyze literature review regarding verification of IFC models and conclude that there are vast research works in this field, but still there are

[1] Open BIM, http://www.buildingsmart.org/openbim/

many open challenges to address. Therefore, we present a need of an approach that can easily be extended, configured and deployed for the dynamic and changing environment having broad spectrum of functionalities for the verification of IFC models.

The main contribution of this paper is about the development of *Semantic BIM Reasoner (SBIM-Reasoner)* which employs semantic technologies to meet the requirements of semantic verification of an IFC model. *SBIM-Reasoner* employs several preprocessors (elaborated in next sections) to build the semantic repository based on RDF [25] from the input IFC model. Once all the triples are generated from the initial data (.ifc file), it uses Stardog[2] to build a knowledge graph for the semantic verification. All types of inference and reasoning mechanisms for the semantic verification are applied over this knowledge graph to meet the requirements of verification, and in addition to discover additional information that is not explicitly stated in the initial data of the IFC model. Our semantic preprocessor uses both forward chaining and backward chaining mechanisms (where appropriate) to build the semantic repository. SPARQL[3] rules (statements and materialization) are applied to enrich the underlying semantic repository with several newer and high level concepts as per demand of regulation texts and verification rules by the end-users. Finally, SPARQL queries are performed over the semantic repository for the verification and code compliance of an IFC model. Once, *SBIM-Reasoner* finds non-compliant objects in the IFC model, it presents them to the end-user. Later in this paper, we also present our analytical results on several online IFC models[4] and also on four IFC models developed at our enterprise. We discuss our experimental finding on different analysis parameters such as number of triplets in the RDF (turtle file) equivalent to IFC model, number of triplets in the semantic model (filtered turtle file) in the Stardog, estimated time taken by the conversion and geometric preprocessor, etc. On the basis of analysis from these parameters, we show encouraging results by several tests on the knowledge graph from the initial version of *SBIM-Reasoner*.

The rest of paper is organized as follows. Section 2 discusses related work. Section 3 presents our *SBIM-Reasoner*, its architecture, subcomponents, and as a semantic service to end-users. This section also presents our statistical analysis of our implemented prototype, empirical results based on various IFC models and highlights various important points. Section 4 concludes the paper and presents future directions

2 RELATED WORK

Over the last few years, many methods and techniques have been proposed for the verification of IFC models. There are three ways for the conformance checking of IFC models as discussed by Pauwels and Zhang [5]. The subsections elaborate each of them.

2.1 Hard Coded Rule Checking

First, we have the hard coded rule checking mechanism for the verification of IFC models, which is similar to the approach adopted by Solibri Model Checker [6]. This tool loads a BIM model, considers rules stored natively in the application and performs rule checking against the BIM for the architectural design validations. This approach is fast as rules are integrated inside the application, but there is no flexibility or customization possible as rules are not available outside the actual application.

The traditional approach of compliance checking is with the *IfcDoc* tool [7] developed by buildingSMART International for generating MvdXML rules through a graphical interface. It is based on the MvdXML specification [22] to improve the consistent and computer-interpretable definition of Model View Definitions as true subsets of the IFC Specification with enhanced definition of concepts. This tool is widely used as AEC specific platform in the construction industry. MvdXML Checker [27] is a great contribution for the automatic verification of IFC models and to detect the non-conformities with the associated 3D visualization, or to provide access to the technical documentation for a given digital model based on its sophisticated contextual information. At our enterprise, we proposed

[2] Stardog triplestore: https://www.stardog.com/

[3] SPARQL http://www.w3.org/TR/rdf-SPARQL-query/

[4] IFC test Data https://github.com/opensourceBIM/TestFiles/tree/master/TestData/data

several extensions and implemented those into a new research prototype. But after these extensions, still we analyze that this traditional approach of verification by the use of MvdXML is very limited and has narrow scope for the verification of IFC models. There are many drawbacks of MVDXML for extracting building views such as: lack of logical formalisms, solely consideration of IFC schema and MVD-based view constructors are not very flexible and dynamic [23]. In addition, major limitations exists such as restricted scope of applying conditions and constraints on several branches of an IFC model, poor geometric analysis of an IFC model, lack of mathematical calculations, support of only static verification of a model, etc. On the other hand, when we practice semantic technologies such as SPARQL, we think their suitability due to wide range of functions, intermediate calculations, and support of dynamic creation of verification rules at ease.

2.2 Query based Rule Checking

The second approach is 'query based rule checking' of an IFC model. In this approach, BIM is interrogated by rules, which are formalized directly into SPARQL queries. As an example, Bouzidi et al. [8] proposed this approach to ease regulation compliance checking in the construction industry. They reformulated the regulatory requirements written in the natural language via SBVR, and then, SPARQL queries perform the conformance checking of IFC models.

2.3 Semantic Rule Checking Approach

The third is a semantic rule checking approach with dedicated rule languages such as SWRL [24], Jess [9] or N3Logic [10]. There are few projects in AEC industries that use this approach for the formal rule-checking, job hazard analysis and regulation compliance checking. Wicaksono et al. [11] built an intelligent energy management system for the building domain by using RDF representation of a construction model. Then, they formulated SWRL rules to infer anomalies over the ontological model. Later, they also developed SPARQL interface to query the results of rules. Pauwels et al. built acoustic regulation compliance checking for BIM models based on N3Logic rules [12]. They use N3logic rules with an ontology to reason whether a construction model is compliant or not with the European acoustic regulations. Another project that was built on the ontological framework for the rule-based inspection of eeBIM-systems was developed by Kadolsky et al. [13]. They used rules to query an IFCOWL ontology that captured a building.

Besides these projects that build an ontology for the IFC, recent years revealed some contributions based on *Semantic Web Technologies*. SWOP-PMO project is one of recent contributions that uses formal methodology based on the Semantic Web standards and technologies [14]. It uses OWL/RDF to represent the knowledge, and SPARQL queries and *Rule Interchange Format* (RIF) to represent the rules. The RDF/OWL representation is not derived from the written knowledge but has to be remodeled in accordance with the rules of OWL/RDF. There are some other works for the semantic enrichment of ontologies in the construction and building domain. Emani et al. proposed a framework for generating an OWL Description Logic (DL) expression of a given concept from its natural language definition automatically [15]. Their framework also takes into account an IFC ontology and the resultant DL expression is built by using the existing IFC entities. Fahad et al. have contributed a framework for mapping certification rules over BIM to enable the compliance checking of the repository through the digital building model [16]. They aimed to align several specialized indexations of building components at both sides, by extending IfcOWL ontology with bSDD vocabulary (i.e., synonyms and description) as enriched IfcOWL ontology to deal with the same abstract concepts or physical objects. Fahad et al. also investigated semantic web approach by using SWRL and traditional approach by the use of IfcDoc tool and analyzed that the semantic web technique represents more global scope with larger visibility of querying for the validation of IFC models [17]. Ontologies play a vital role for the rule based semantic checking, therefore in the next subsection to mention some of the important ontologies in the IFC domain.

2.3.1 Ontologies in the IFC domain

To achieve the benefits of ontologies, there are many efforts to build an ontology for the IFC construction industry. One of the outcomes can be seen as an IFC-based Construction Industry Ontology and Semantic web services framework

[18]. With simple reasoning built over the ontology, their information retrieval system could query the IFC model into XML format directly. The BuildingSMART Linked Data Working Group has developed IfcOWL ontology to allow ex-tensions towards other structured data sets that are made available using semantic web technologies [19]. There are many versions of IfcOWL ontology since the work has been started. We have been work-ing on an ontology IFC4_ADD1.owl that came on 25 Sept. 2015. We have enriched this ontology with English-French and IFC vocabulary (synonyms, descriptions, etc.) from bSDD semantic data dictionary in our research project where we map regulatory text and certification rules over BIM [16]. In addition, we assigned concepts of IfcOWL ontology with Global Unique Identifier (GUID) to serve as a unique language-dependent serial number from the bSDD. There are some other ontologies as well such as the ontology defining the core concepts of a building named Building Topology Ontology (BOT) [20] and the ontology for CAD Data and Geometric Constraints named OntoBREP [21].

3 SEMANTIC BIM REASONER

This section presents our research and development towards building *Semantic BIM Reasoner*. The following subsections elaborate it's various aspects.

3.1 SBIM-Reasoner Architecture

Semantic BIM Reasoner (*SBIM-Reasoner*) deploys many preprocessors for building semantic repository for the verification of IFC models (see Figure 1). Primarily it has three pre-processors, i.e., IFC to RDF Converter, Geometry Extractor, and Semantic Preprocessor which has further sub-components named IFCOWL Ontology sub-graph, SPARQL Rules, SPARQL Queries and TripleStore.

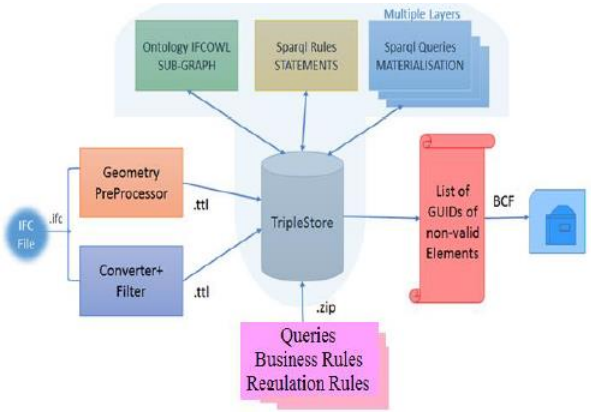

Figure 1. Top level Architecture of *SBIM-Reasoner*

3.1.1 IFC-to-RDF Converter+Filter

It is necessary for the semantic reasoner to con-vert IFC into RDF for building the semantic repository. *IFC-to-RDF* is a set of reusable Java component that allows parsing IFC files and converts them into RDF graphs. Our system deploys modified version of IFC-to-RDF conversion plug-in provided by Pauwels & Oraskari [26]. After conversion, underly-ing RDF acts as a foundation stone to execute all the verification rules. Therefore, we did filtration to get only relevant triplets from the IFC model. Generally there are two ways to get filtered model. First to get full RDF equivalent of IFC model and then apply *SPARQL Construct query* to get small graph of only wanted IFC classes. In this approach we found over-head of creating full graph and then extracting a sub-graph. Second, which we adopted is to integrate *IF-Then-Else* statements inside the code of Pauwels to filter unwanted elements like IFC classes {Person, Address, MaterialList, SwitchingDeviceType, ColourRgb, etc.}. By this filtration, we have noted that we got filtered RDF model which is 10 times smaller in size as compared to full RDF (equivalent IFC) model. Table 1 shows the comparison between RDF files, i.e., RDF equivalent IFC and Filtered RDF.

3.1.2 Geometry Preprocessor

There are two geometry render engine plugins available with the *BIM Server* named *IFCOPENSHELL* and *IFC Engine DLL*. These are helpful to extract geometry data about the IFC objects. The outputs of this preprocessor are the RDF triplets which are formed from the extracted geometry data of relevant IFC objects. Table 2 shows the statistics of IFC objects present in the IFC models.

Table 1. Comparison between RDF Files (Equivalent IFC vs. Filtered RDF)

IFC model	IFC Size Mb	RDF Size (Mb)		No of Triplets	
		Equivalent IFC	RDF – Filtered ttl	RDF Equivalent IFC	RDF – Filtered ttl
HITOS	62.5	404.6	16.6	5,054,202	205,631
AC9R1-Haus	4.4	39.1	0.856	490,961	10,982
BIM_EM	1.9	21.3	0.149	275,927	2,138
Candidat-23_04	28.9	56.6	11.4	694,709	132,115
Chanteloup	17.1	63.2	12.3	807,250	142,668
LcD	32.4	130.7	11.4	1627,750	132,845
Bat_CSTB	14.9	79.5	1.0	1047,216	13,973
Maquette Test Checker	11.2	122.6	7.7	1,572,792	100,230
HAixFlowCtrl	13.1	155.6	1.7	1,970,366	21,549
Liberty_V3	12.2	137.2	6.6	1,760,207	86,545

Table 2. Statistics of IFC objects in IFC models

IFC model	Door	Wall	Wall(sd)	Window	Space	Stair	Slab	Covering	Open-El	FlowSeg	FlowTrml	FlowCtrl	Railing
HITOS	198	4	842	226	243	14	363	0	519	0	67	0	6
AC9R1-Haus	25	0	60	35	26	4	35	0	67	0	0	0	16
BIM_EM	9	0	33	48	17	0	16	0	58	0	0	0	0
Candidat	204	0	392	434	17	0	43	0	649	0	0	0	0
Chanteloup	153	0	328	77	166	3	46	0	228	0	0	0	21
LcD	256	0	634	74	360	5	6	0	424	0	0	0	63
Bat_CSTB	65	0	208	102	59	0	17	0	176	0	0	0	5
HAixFlowCtrl	0	0	0	0	0	0	0	0	0	0	0	1647	0
Liberty_V3	85	0	599	136	68	5	12	47	321	2	15	5	25

*Wall(Sd)=StandardCaseWall, Open-El=OpeningElement, FlowSeg=FlowSegment, FlowTrml=FlowTerminal, FlowCtrl= FlowController

3.1.3 IFCOWL ontology sub-graph

As the standard IFCOWL ontology has a very large set of IFC elements, therefore, we deal with the sub-graph to achieve better processing and querying performance.

3.1.4 SPARQL Rules - Statements

We have created a large set of SPARQL rules, i.e., statements. In fact, these statements are shortcuts overs the long chain of triplets to enable simplicity. For instance, we created *In_Storey* shortcut over the RDF triplets via relatedObjects and relatingObjects between IFC elements (see Figure 2). Likewise *Boundary* statement is created as a shortcut over the RDF triplets via relatingSpace and relat-edBuildingElement. These statements promote readability, understandability and enable simplicity when creating SPARQL rules and queries. Otherwise the chain of triplets makes things complex and ambiguous.

```
[] a rule:SPARQLRule ;
        rule:content """
                PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                PREFIX ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
                PREFIX rule: <tag:stardog:api:rule:>
                PREFIX list: <https://w3id.org/list#>
                PREFIX express: <https://w3id.org/express#>

                IF {
                        ?agr ifcowl:relatedObjects_IfcRelDecomposes ?element1 .
                        ?agr ifcowl:relatingObject_IfcRelDecomposes ?element2 .
                } THEN {
                        ?element1 ifcowl:inStorey ?element2  .
                }
""" .
```

In_Storey Equivalent Shortcut in Stardog

```
[] a rule:SPARQLRule ;
        rule:content """
                PREFIX ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>

                IF {
                        ?rel ifcowl:relatingSpace_IfcRelSpaceBoundary  ?space .
                        ?rel ifcowl:relatedBuildingElement_IfcRelSpaceBoundary  ?element .
                } THEN {
                        ?space ifcowl:boundary ?element  .
                }
""" .
```

Boundary Equivalent Shortcut in Stardog

Figure 2. Examples of Statements over triplets

3.1.5 SPARQL Queries - Materialization

During the analysis of rules specification, we come across various types of vocabulary (introduced by

regulatory texts) during building code compliance application. This vocabulary is composed of high level concepts present in business rules and regulation texts which are familiar by the stakeholders of BIM. There are two methods to build such vocabulary of newer high-level concepts, i.e.; via forwarding chaining and/or backward chaining. Based on the SPARQL rules, we have built SPARQL queries to introduce high level concepts based on the primary IFC vocabulary by using both forward and backward chaining where applicable. Figure 3 shows high level concepts *'circulationHorizontale'* and *'Degagement'* in our case study of building French code compliance via forward chaining. In these examples we have used our defined *inStorey* and *boundry* concepts along with the pre-defined IFC owl terminologies such as *IfcOpeningElement*, *IfcSpace*.

```
CONSTRUCT {?s1 a ifcowl:circulationHorizontale}
{
  SELECT ?s1 ( COUNT(?s2) AS ?cnt )
WHERE {
    ?s1 a ifcowl:IfcSpace ; ifcowl:boundary ?element ; ifcowl:inStorey ?storey .
    ?element rdf:type ?type .
    ?s2 a ifcowl:IfcSpace ; ifcowl:boundary ?element ; ifcowl:inStorey ?storey .
    VALUES ?type { ifcowl:IfcDoor ifcowl:IfcOpeningElement } .
    FILTER ( ?s1 != ?s2 )
        }
  GROUP BY ?s1
HAVING (?cnt > 2)
}
```

```
CONSTRUCT {?e a ifcowl:degagement}
WHERE {
                ?e a ?type .
   VALUES ?type { ifcowl:circulationVerticale ifcowl:circulationHorizontale
ifcowl:IfcDoor }
}
```

Figure 3. Examples of Forward Chaining SPARQL Queries

Backward chaining consists of ontology statements that align IFC concepts with regulatory concepts, whereas forward chaining consists of insert statements that create supplementary triplets. Forward chaining is a good at an implementation stage to save memory and CPU resources. From the ma-chine point of view, backward chaining is processed each time a semantic query is submitted whereas forward chaining is executed each time the data changes. At this stage, this choice is a compromise between effective queries (forward chaining is more appropriate for complex and numerous queries) and model update frequency (backward chaining is more appropriate when data changes frequently). We can even say that it is a compromise between the amount of triplet (considering triplet generated by forward chaining statements) and the ontology complexity. Therefore, we have mixed both approaches Backward and Forward chaining to provide the optimal setting that minimizes response time and maximizes ontology consistence. With the help of these techniques, we have simplified several IFC patterns such as classifications, predefined types, properties, geometry, topology, etc.

3.1.6 TripleStore - Stardog

Although IFC is an open standard; its complex nature makes information retrieval difficult from an IFC model as the size of IFC model grows. Therefore, we have used Stardog as a triplestore to build our semantic model. Querying semantic model is faster and gives a good runtime. When the application starts, an end-user provides an IFC model and the set of SPARQL queries which are the verification rules for checking code compliance of desired IFC model. As a result, our system converts IFC file into filtered RDF model. It loads that converted-filtered IFC equivalent RDF into stardog. After triplets concerning geometry are added to capture geometrical information in the triplestore. Then the semantic model is enriched with IFCOWL basic vocabulary, i.e., sub-graph of IFC ontology. Then, it adds SPARQL rules into the triplestore. Finally, it executes our project specific forward chaining SPARQL queries which creates high level vocabulary and builds further RDF graphs over the existing triplets. With reference to above examples, our semantic preprocessor is illustrated in Figure 4 based on the basic IFC vocabulary, shortcuts and constructs.

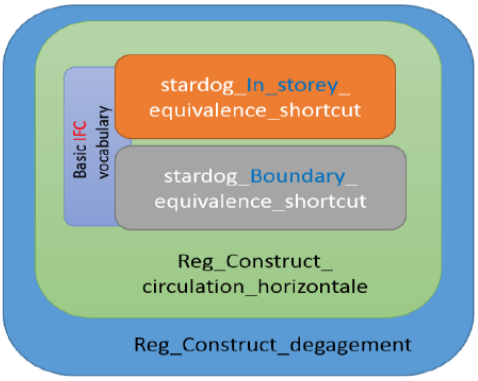

Figure 4. Example of Construction of Semantic Preprocessor

3.1.7 End-User Queries

Stardog provides fast access to triplets to fetch data to validate IFC models. All the end-user verification queries are executed on the top of final stardog triple store which is built successively by our reasoner. For example if an end-user needs to check whether a Room in the IFC model is accessible by a wheelchair, then it can be done easily by using above explained construct_degagement where the value of IfcDoor must be equal or greater than 90 cm. An end-user may apply SPARQL ASK and DESCRIBE queries to retrieve relevant information regarding the verification rules. Instead of using IfcDoc tool where there is no intermediate state and no explanation for the reason of non-compliance, we use SPARQL DESCRIBE Queries. The SPARQL DESCRIBE query does not actually return resources matched by the graph pattern of the query, but an RDF graph that "describes" those resources. It is up to the SPARQL query service to choose what triples are included to describe a resource. Therefore, SPARQL queries serve best by concatenating desired triplets for building verification rules to check the code compliance.

3.2 SBIM-Reasoner as a semantic service

We have developed *SBIM-Reasoner* as a semantic service inside a KROQI platform[5]. As it is developed especially for the *French building code compliance*, therefore our web interface is also in French targeting French community. When the application starts, an end-user has to configure IFC input model by clicking under the synchronization button on the very first tab 'Maquette'. Then an end-user selects the set of rules to be verified on this input model by selecting/browsing their set of rules on the second tab 'Protocoles'. Then our web service starts by calling semantic reasoner which computes the set of rules and redirects to the result page 'Résultats'. Each of the rule is highlighted as green or red color depending on its status of compliance (see Figure 5).

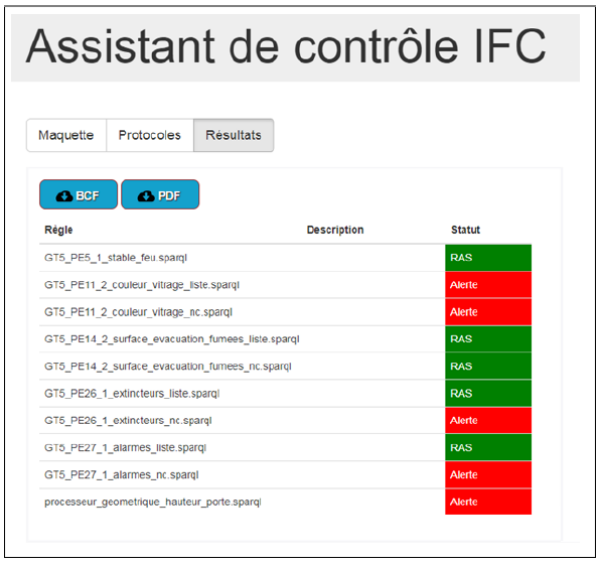

Figure 5. Result Page containing status of Verification Rules

When *SBIM-Reasoner* has detected non-compliant objects (in case of red status), an end-user can further analyze them by clicking on the corresponding row. It fetches and displays the list of IFC non-compliant objects containing Name, GUID and Type of each IFC object (see Figure 6). One can also export PDF and BCF files to analyze their results in detail.

Figure 6. List of non-compliant IFC objects

3.3 Testing IFC Models

We have used several IFC models from the online repository (see url above) to test our *Semantic BIM Reasoner*. These IFC models vary in size, number of IFC objects, free spaces, etc. We have

[5] https://svc-bimsemchecker.dev.coplus.fr/CheckerService-/v2/ui?file_id=test_20180302_NR_LibertyLoft_OK

also used four IFC models developed at our CSTB enterprise named Bat_CSTB (14.9 MB), HAixFlowCtrl (13.1 MB), Maquette Test Checker (11.2 MB) and Liberty (12.2 Mb). Above in Tables 1 and 2 we have shown statistics about these IFC models. We have also measured time taken by the *Converter Preprocessor* that does conversion and filtration to produce the initial RDF semantic model on a 'quad core i5 CPU at 2,5Ghz' machine. In addition, we have also measure time taken by the *Geometry Preprocessor* that extracts geometry data of our desired IFC elements (ref. Table 2) from the IFC models. Figure 7 shows the estimated time taken by the conversion and geometry preprocessors. When we see the time graph, we observe that *SBIM-Reasoner* took less than a minute by both the preprocessors to achieve their objectives when the size of IFC model is under 15 MB. But when the size of IFC model is 62 MB (in case of huge IFC model Hitos) then it took almost 3.4 minutes to convert and filter, and more than 5 minutes to extract geometry data. Here, we also mention that although preprocessors took time to build semantic model at the first time, but querying for the verification of IFC models are processed fast and a good runtime is achieved. On the other hand on traditional IFC model, it takes much time to verify each of the individual rule. In addition, we are not able to execute all types of rules as per our desire due to narrow scope of IFC tools available online. This is only with the semantic model that we are flexible enough to fetch any triplets and build rules according to our will for the verification of IFC models. On the basis of this analysis, we conclude that semantic model serves best for the verification of IFC models. We also revealed encouraging results via several tests from the initial version of *SBIM-Reasoner*.

4 CONCLUSIONS

There are many techniques for the automatic verification of IFC models, but, still there are many open challenges. In this paper, we have presented a semantic approach that can easily be extended, configured and deployed for the dynamic and changing environment having broad spectrum of functionalities for the verification of IFC models. We presented *SBIM-Reasoner* that builds semantic model by con-verting IFC model into RDF and also extracting geometry data as a set of triplets. Then, it loads this semantic RDF data into Stardog triplestore, and enriches the primary semantic model with our defined SPARQL Rules (shortcuts overs long chain of triplets) and Queries (regulation and business rules). End-user queries are formalized as SPARQL queries which are executed on the top of this final triplestore. *SBIM-Reasoner* responds their status of compliance along with the BCF representation of non-compliant IFC objects. We demonstrated several test models on *SBIM-Reasoner* and presented its efficiency and efficacy with empirical results. We conclude that the semantic model based on the semantic web technology is a good compromise between development efforts and opportunities. The graphical representation of RDF allows rules to be more intuitive and more efficient to reason and execute. Concatenation of triplets allows flexibility of making wide range of verification rules with condition and constraints at ease. SPARQL has a global scope with larger visibility of querying with the built-in functions and support of intermediate calculations for the validation of IFC models.

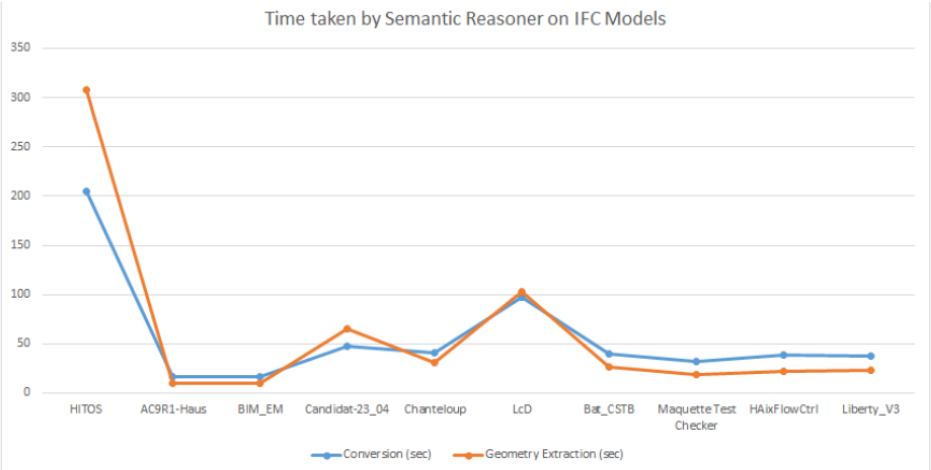

Figure 7. Time taken by preprocessors (Converter and Geometry Extractor)

References

[1] Rebekka, V., Stengel, J. & Schultmann, F. 2014. "Building Information Modeling (BIM) for existing buildings—Literature review and future needs." Automation in construction, vol. 38, pp. 109-127.

[2] Eastman, C., Teicholz, P., Sacks, R. & Liston, K. 2008. "BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors", Hoboken, New Jersey, Wiley.

[3] Thein, V. 2011. Industry Foundation Classes (IFC), BIM Interoperability Through a Vendor-Independent File Format, A Bentley White Paper, September 11.

[4] Ismail, A.S., Ali, K.N. & Iahad, N.A. 2017. "A Review on BIM-based automated code compliance checking system," 2017 International Conference on Research and Innovation in Information Systems (ICRIIS), Langkawi, pp. 1-6.

[5] Pauwels, P. & Zhang, S. 2015. "Semantic Rule-Checking for regulation compliance checking: An overview of strategies and approaches". Proc. of the 32nd CIB W78 Conference, Netherlands.

[6] Khemlani, L. 2009. "Solibri model checker", AECbytes Product Review 31st March 2009.

[7] IfcDoc Tool, available at: http://www.buildingsmart-tech.org/specifications/ specification-tools /ifcdoc-tool/ifcdoc- help-page-section/IfcDoc.pdf, 2012.

[8] Bouzidi, K.R., Fies, B., Faron-Zucker, C., Zarli, A. & Thanh, N.Le. 2012. "Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry". *Future Internet*. 4 (3). pp. 830-851.

[9] Friedman-Hill, E. 2003. "Jess in Action: Rule Based Systems in Java". Manning Publications. ISBN 1-930110-89-8.

[10] Berners-Lee, T.I.M., Connolly, D.A.N., Kagal, L., Scharf, Y. & Hendler, J.I.M. 2008. "N3Logic: A logical framework for the World Wide Web", Theory and Practice of Logic Programming. 8 (3), doi:10.1017/S1471068407003213.

[11] Wicaksono, H., Dobreva, P., Häfner, P. & Rogalski, S. 2013. "Ontology development towards expressive and reasoning-enabled building information model for an intelligent energy management system". Proc. of the 5th KEOD, pp. 38-47. SciTePress,

[12] Pauwels, P., Deursen, D.Van, Verstraeten, R., Roo, J.De, Meyer, R.De, De-Walle, R.Van, & Van-Campenhout, J. 2011. "A semantic rule checking environment for building performance checking". Automation in Construction. 20 (5). pp. 506-518.

[13] Kadolsky, M., Baumgärtel, K. & Scherer, R.J. 2014. An ontology framework for rule-based inspection of eeBIM-systems. Procedia Engineering. vol. 85, pp. 293-301.

[14] Josefiak, F., Bohms, H., Bonsma, P. & Bourdeau, M. "Semantic product modelling with SWOP's PMO, eWork and eBusiness in AEC", pp.95-104

[15] Emani, C.K., Ferreira Da Silva, C., Fiès, B., Ghodous, P. & Bourdeau, M. 2015. "Automated Semantic Enrichment of Ontologies in the Construction Domain". Proc. of the 32nd CIB W78 Conference, Netherlands,

[16] Fahad, M., Bus, N. & Andrieux, F. 2016 "Towards Mapping Certification Rules over BIM", Proc. of the 33rd CIB W78 Conference, Brisbane, Australia, 2016.

[17] Fahad, M., Bus, N. & Andrieux, F. 2016, "SWRL Towards Mapping Certification Rules over BIM", Proc. of the 33rd CIB W78 Conference, Brisbane, Australia

[18] Zhang, L. & Issa, R.R. 2011. "Development of IFC-based Construction Industry Ontology for Information Retrieval from IFC Models", International Workshop on Computing in Civil Engineering, Netherlands, Vol. 68.

[19] Terkaj, W. & Pauwels, P. 2014. "IfcOWL ontology file for IFC4", Available at: http://linkedbuildingdata.net/resources/IFC4_ADD1 .owl

[20] Rasmussen, M.H., Schneider, G.F. & Pauwels, P. Building Topology Ontology (BOT), https://github.com/w3c-lbd-cg/bot

[21] Perzylo, A.C., Somani, N., Rickert, M. & Knoll, A. 2015. An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions. In proceedings of IROS'15: 4197-4203E.

[22] Chipman, T., Liebich, T. & Weise, M. 2012. "mvdXML specification of a standardized format to define and exchange MVD with exchange requirements and validation Rules", version 1.0.

[23] Mendes de Farias, T., Roxin, A. & Nicolle, C. 2016. "A Semantic Web Approach for defining Building Views", buildingSMART Summit Jeju, Korea.

[24] Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof B. & Dean, M. 2004. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML".

[25] Brickley, D., Guha, R.V. & McBride B. 2004. "RDF vocabulary description language 1.0: RDF Schema". W3C Recommendation 2004.

[26] Pauwels, P. & Oraskari, J., "IFC-to-RDF Converter" https://github.com/IDLabResearch/IFC-to-RDF-converter

[27] MvdXMLChecker, available at: https://github.com/opensourceBIM/mvdXMLChecker