Conformance Checking of IFC Models via Semantic BIM Reasoner

Muhammad Fahad, Nicolas Bus
Centre Scientifique et Technique du Batiment (CSTB), 06560 Valbonne, France
{fahad.muhammad, Nicolas.bus}@cstb.fr

Abstract. In the Building Information Modeling world, compliance checking of IFC models is an immense challenge. Many approaches such as Hard Coded Rule Checking, Query based Rule Checking, Rule Checking Approaches via dedicated rule languages (SWRL, Jess, N3Logic, etc., are being contributed to fulfil these requirements. In this paper, we present our research and development of Semantic BIM Reasoner (SBIM-Reasoner) tool for measuring and ensuring conformance of IFC models. SBIM-Reasoner implements multi-threading approach for embedding semantic querying and rule based approach for developing French building code compliance system. Various pre-processors (*IFC2RDF Converter, Geometry Extractor, Rule Evaluator*, etc.) run in different execution threads to build the underlying semantic repository providing faster rule checking and information retrieval from the triple store. When it finds non-compliant objects in the IFC model, it presents them to the end-user. With the knowledge graph over triplets, we have the freedom of extending our semantic IFC model, creation of newer vocabulary and formation of newer rules, concatenation of triplets to build rules with conditions and constraints over IFC data, dynamic reasoning over the triplets based on the initial data of IFC model, etc. We present encouraging results by various tests which were build using online IFC test models of various sizes and designs to test the performance of SBIM-Reasoner.

1. Introduction

Buildings must conform to the building regulations (commonly known as building codes) which are the policies and guidelines to identify the least acceptable level of security, hazards, accessibility, general welfare, etc. (Rebekka and Schultmann, 2014). Through building codes and standards, organizations accomplish their ultimate aim to guard public health, safety and general welfare as they relate to the construction and occupancy of buildings. For an example earthquake-resistant building codes verify building structures so that they can withstand during the seismic activity. The purpose of verifying models is to align several specialized indexations of building components at both sides, assuming that they deal with the same abstract concepts or physical objects, but according to their distinct representation prisms (Eastman et al., 2008). Building Code is vital to detect non-compliance elements in the IFC model (Thein, 2011) and to guarantee its quality and reliability in the entire life-cycle of Building Information Modeling (BIM). Many approaches and tools for the compliance checking of IFC models are being contributed to fulfil these requirements such as; *Hard Coded Rule Checking*, *Query based Rule Checking* and *Rule Checking via Dedicated Rule Languages* (Pauwels & Zhang, 2015). During our research for the French building codes compliance system, we investigated deeply whether which approach divulges more recompenses for the compliance checking of building codes. According to our exploration, a *hybrid approach* based on semantic querying for the information retrieval and building rule evaluator based on the semantic rule engine produces good results. Therefore, we have implemented a hybrid approach for French building code compliance system named *Semantic BIM Reasoner (SBIM-Reasoner)* which reasons compliance of building codes via semantic reasoning over RDF Triples. Building codes are implemented as SPARQL (2013) queries which run on the semantic repository to verify correctness and consistency of IFC objects in the building model. We use both forward chaining and backward chaining mechanisms (where appropriate) to build our semantic reasoner.

Several newer concepts, SPARQL rules (statements and materialization) are applied to enrich the underlying semantic repository as per demand of verification rules. Use of forward chaining mechanisms support the dynamic semantic verification for the future checks. We build several test cases comprise of different queries on different sizes of IFC models to evaluate our semantic approach. We discuss our experimental findings on many different analysis parameters; such as *number of RDF triples* in the RDF (turtle file) equivalent to IFC model, number of RDF triples in the semantic model (filtered turtle file) in the triple-store, estimated time taken by the *conversion pre-processor* and *geometric pre-processor*, etc. From the initial results, we conclude that SPARQL queries are flexible for retrieving data and perform validation in an optimized way giving better run-time as compared to the traditional approaches.

The rest of paper is organized as follows. Section 2 presents related work. Section 3 presents our research and development of Semantic BIM Reasoner and its different sub-components. Section 4 presents our experimental analysis via using online building models. Section 5 concludes this paper and shows our future directions.

2. Related Work

Many approaches and tools are being contributed to fulfil these requirements. Primarily Hard Coded Rule Checking mechanisms via MVDs are used where building codes are assimilated inside the application for the conformance checking of IFC models (Zhang et al., 2014). Tools such as Solibri Model Checking (Khemlani, 2009), IfcDoc (2012), etc. are contributed are examples of this approach. 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 (Mendes de Farias et al., 2016). The subset of the IFC schema needed to satisfy one or many Exchange Requirements of the AEC industry is called Model View Definition (MVD) and mvdXML is an open standard used to publish the concepts and associated rules (Chipman et al., 2016). It can be used with the BIM Server or IfcDoc tool developed by the buildingSMART International to read and write mvdXML and to provide a graphical user interface for defining all content within mvdXML. Due to the limitation of MVD and mvdXML, semantic web technologies have seen as an option which is regarded as a good compromise between development efforts and possibilities. Therefore Query based Rule Checking mechanisms are emerged, where BIM model is interrogated via conformance rules that are formalized directly into SPARQL queries. The contribution from Bouzidi et al. (2012) is regarded as Rule-checking by querying. They transform building model into RDF, create rules as SPARQL queries, and finally interpret query result to visualize the result. Recently Rule Checking Approaches via dedicated rule languages were developed for the rule-based inspection of IFC models for the conformance checking. Most recent contributions use rule languages such as; Wicaksono et al. (2013) based on SWRL rules (Horrocks, 2004), Pauwels et al. (2001) based on N3Logic rules (Berners-Lee, 2008) and M. Kadolsky et al. (2014). We have implemented a hybrid approach (SPARQL + Rule Engine) for French building code compliance system named Semantic BIM Reasoner (*SBIM-Reasoner*) which reasons compliance of building codes via semantic reasoning over RDF Triples. Our work is similar to these proposals as we also make advantage of semantic web technologies, but, we also consider geometric aspects, with focus on optimization and performance. We presented first version of our developed semantic reasoner which was without multi-threading approach (Fahad et al., 2017). This paper presents our optimized solution using multi-threading approach. We build several test cases comprise of different queries on different sizes of IFC models to evaluate our semantic approach. The only overhead is the conversion from an *IFC to RDF* and then storage of RDF triples into triple store which takes time. In a long run, once the

triplestore is loaded with the data, querying is much faster to validate IFC models and detect inconsistent non-compliant IFC elements in the building model.

3. Multi-Threaded Semantic Approach for the Conformance Checking of BIM

The main contribution in this paper is about the research and development of *Semantic BIM Reasoner (SBIM-Reasoner)* tool for measuring and ensuring conformance of IFC models. The idea is to formulize building code rule as a SPARQL query and then execute it on the building model, which is formulated as triplestore via forward/backward chaining mechanisms, to check the possibility of non-compliant elements in the building model. Figure 1 illustrates the top level architecture of SBIM-Reasoner. *SBIM-Reasoner* implements multi-threaded approach for embedding semantic querying and rule based approach for developing French building code compliance system. The main thread gets desired IFC models as input and triggers the *Parallel Execution* of *Pre-Processsors* for the execution of different tasks. 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. Finally rule evaluator executes queries over semantic repository to checking and report the conformance report of IFC objects.

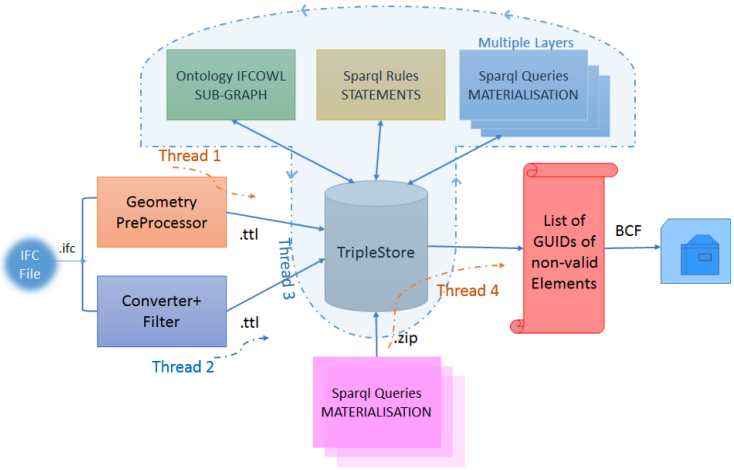

Figure 1: Top level architecture of SBIM-Reasoner

3.1 IFC2RDF Conversion Pre-processor

First thread executes conversion pre-processor to build the initial semantic repository based on RDF triples equivalently converted from an IFC model. Figure 2 shows partial RDF triples of an IfcDoor object. *IFC-to-RDF* is a set of reusable Java component that allows parsing IFC files and converts them into RDF graphs. Our implementation deploys modified version of IFC-to-RDF conversion plug-in provided by Pauwels & Oraskari (2012). After conversion, underlying RDF semantic model acts as a foundation stone to execute all the verification rules. We also need to apply filtration to get an RDF equivalent compact triple file to avoid several IFC elements, such as *Person*, *Address*, *Material-List*, etc. Therefore, the same thread did filtration to get only relevant RDF triples from the IFC model.

3.2 Geometric Pre-processor

Second thread executes geometric pre-processor to extract geometric related data from an IFC model. It employs *Geometry Render Engine* to extract geometric information and stores them

as RDF triples in the semantic repository. Figure 2 illustrates the output triples corresponding to bounding box values of an IfcDoor object. Mainly 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 triples which are formed from the extracted geometry data of relevant IFC objects. These two pre-processors are executed in parallel to gain time. The advantage of their parallel execution is that none of the threads wait for each other, their functionality is different so they can efficiently utilize the server processor to achieve their desired tasks. These threads propagate their results to the next thread. Once all RDF triples (i.e., Filtered RDF file of an IFC model and RDF triples corresponding to Geometry data) are generated by first and second threads, third thread gets and loads these RDF triples into the triple store for the fast querying, searching, and analyzing of RDF triples.

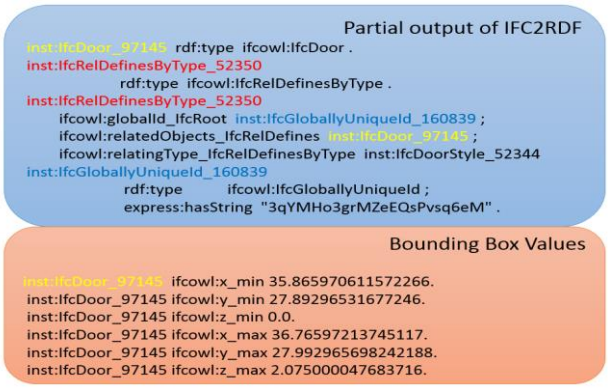

Figure 2: RDF triples corresponding to an IfcDoor object

3.3 Semantic Pre-processor

Third Thread executes to build the semantic repository (i.e., triplestore) with the output of previously discussed two pre-processors. It waits till it is triggered by the outputs of Converter thread and Geometry Extractor thread. It will not start until both the threads finish their execution completely and generate RDF triples. It uploads all the RDF triples to the triplestore. All types of inference and reasoning mechanisms for the semantic verification are applied over this semantic RDF repository to meet the requirements of compliance checking, and in addition to discover additional information that is not explicitly stated in the initial data of the IFC model. We have integrated both forward chaining and backward chaining mechanisms (where appropriate) to build our semantic repository. Several newer concepts, SPARQL rules (statements and materialization) are applied to enrich the underlying semantic repository as per the demand of verification rules. Use of forward chaining mechanisms support the dynamic semantic verification for the future checks. The following are its sub-components.

TripleStore – Stardog. Although IFC is an open standard; its complex nature makes information retrieval difficult from an IFC model when the size of IFC model grows. Therefore, we have used Stardog as a triplestore to build our semantic model so that querying semantic model is faster and gives a good run-time. 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 an IFC file into filtered RDF model. It loads that converted-filtered IFC equivalent RDF into stardog. After RDF triples 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.

IfcOWL ontology sub-graph. As the standard IfcOWL ontology (Terkaj & Pauwels, 2014) has a very large set of IFC elements, therefore, we deal with the sub-graph to achieve better processing and querying performance.

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 '*intersects'* shortcut over the RDF triplets corresponding to bounded box values of IFC elements (see Figure 3). These statements promote readability, understandability and enable simplicity when creating SPARQL rules and queries. Otherwise the chain of triplets make things complex and ambiguous.

```
PREFIX rule: <tag:stardog:api:rule:>
[] a rule:SPARQLRule ; rule:content """
      IF {
            ?a ifcowl:x_min ?a_xmin . ?a ifcowl:x_max ?a_xmax .
            ?a ifcowl:y_min ?a_ymin . ?a ifcowl:y_max ?a_ymax .
            ?a ifcowl:z_min ?a_zmin .  ?a ifcowl:z_max ?a_zmax .
            ?b ifcowl:x_min ?b_xmin . ?b ifcowl:x_max ?b_xmax .
            ?b ifcowl:y_min ?b_ymin . ?b ifcowl:y_max ?b_ymax .
            ?b ifcowl:z_min ?b_zmin . ?b ifcowl:z_max ?b_zmax .
            FILTER((?a_xmin <= ?b_xmax && ?a_xmax >= ?b_xmin) &&
            (?a_ymin <= ?b_ymax && ?a_ymax >= ?b_ymin) &&
            (?a_zmin + 0.1 <= ?b_zmax && ?a_zmax - 0.1 >= ?b_zmin)) .
        }  THEN {   ?a ifcowl:intersects ?b  . }     """ .
```

Figure 3: Intersect relation between two IFC objects

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 *'intersects'* in our case study of building French code compliance via forward chaining. 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 machine 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.

3.4 Rule Evaluator - Compliance Checker

Fourth Thread executes rule evaluator also called compliance checker as it performs the fundamental task of evaluating building model against building codes. It waits till it is triggered

by the outputs of third thread that builds semantic repository to fast access triples to fetch data and validate IFC models. It performs SPARQL queries over the semantic repository for the verification and code compliance of an IFC model. 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. Figure 5 shows an example of SPARQL query to detect whether an 'Alarm' is installed at some space in the building. This query is based on semantic rules *ifcowl:in and ifcowl:inStorey* which are also shown below the query. Once, it finds an IFC model which is not instatlled with an Alarm (in this example) or non-compliant objects in the IFC model (in other cases), it presents them to the end-user as inconsistent elements in the building model. With the knowledge graph over RDF triples, we have the freedom of extending our semantic IFC model, creation of newer vocabulary and formation of newer rules, concatenation of RDF triples to build rules with condition and constraints over IFC data, dynamic reasoning over the RDF triples based on the initial data of IFC model, etc.

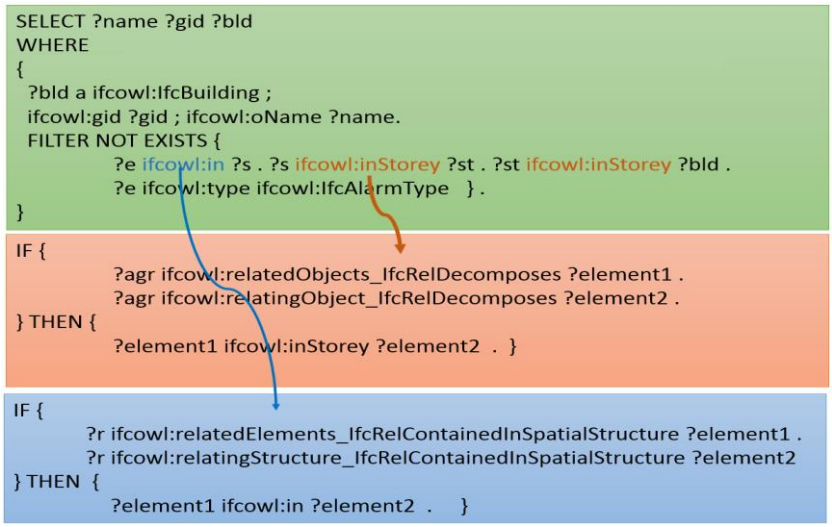

Figure 4: SPARQL Query build on several shortcuts

4. Implementation and Testing

The following sub-sections elaborate implementation and testing details of SBIM-Reasoner.

4.1 SBIM-Reasoner as a Semantic Service

We have developed *SBIM-Reasoner* as a semantic service inside a KROQI platform. 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 5a). When *SBIM-Reasoner* detects non-compliant objects (in case of red status), an end-user can further analyze them by clicking on the corresponding row. It displays the list of IFC non-compliant objects containing Name, GUID and Type of each IFC object (see Figure 5b). One can also export PDF and BCF files to analyze their results in detail.

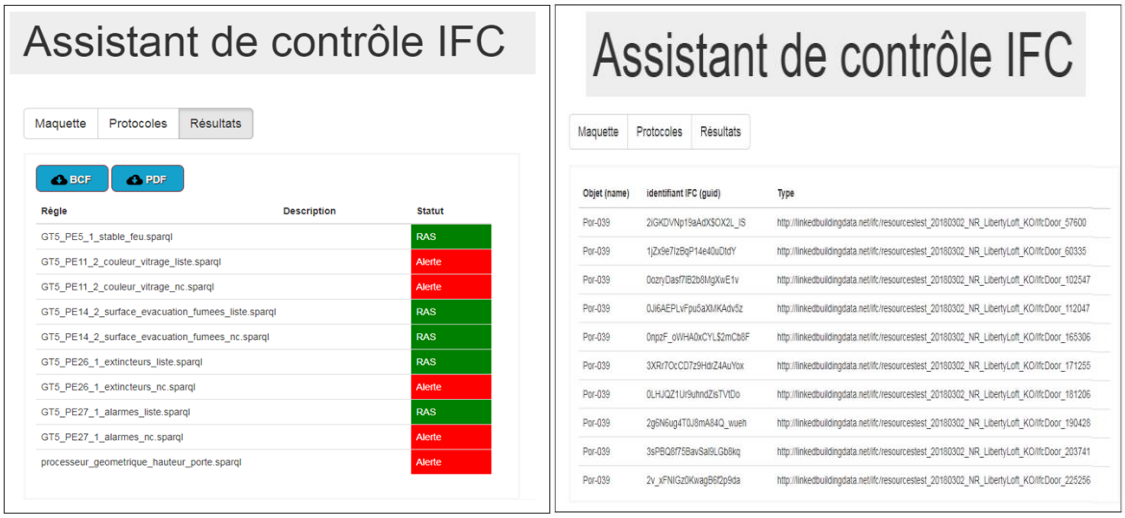

Figure 5: (a) Final output of SBIM-Reasoner, (b) Showing details of inconsistent IFC objects

4.2 Experimental Finding on SBIM-Reasoner

We deployed SBIM-Reasoner service on a cluster kubernetes of 3 nodes (1 node = 1 VM), where each node has 4 vCPUs, 26 GB memory and type of Intel(R) Xeon(R) CPU @ 2.30GHz. We choose several IFC models (see Table 1.) from the online repository to test our *Semantic BIM Reasoner* such as; Chanteloup (17 Mb), AC9R1-Haus (4 Mb), BIM-EM (2 Mb), Candidat (29 Mb), HITOS (63 Mb), and LcD (32 Mb), etc.

Table 1: Size of an IFC model and corresponding number of triples

IFC model	**IFC Size (Mb)**	**No of Triplets**	
		RDF Equiv IFC	**RDF – Filtered ttl**
HITOS	62.5	5,054,202	205,631
AC9R1-Haus	4.4	490,961	10,982
BIM_EM	1.9	275,927	2,138
Candidat-23_04	28.9	694,709	132,115
Chanteloup	17.1	807,250	142,668
LcD	32.4	1627,750	132,845
Bat_CSTB	14.9	1047,216	13,973
Maq Test Checker	11.2	1,572,792	100,230
HAixFlowCtrl	13.1	1,970,366	21,549
Liberty_V3	12.2	1,760,207	86,545

These IFC models vary in size, number of IFC objects, free spaces, etc. We have 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). Tables 1 shows the size of these IFC models, the number of triples in the RDF equivalent to the original IFC model and the filtered model by IFC2RDF pre-processor. We took many different analysis parameters such as *number of RDF triples* in the RDF (turtle file) equivalent to IFC model, number of RDF triples in the semantic model (filtered turtle file) in the triple-store, estimated time taken by the *conversion pre-processor* and *geometric pre-processor*, etc. We have measured time taken by different *preprocessors* that produce desired output in Table 2. Our first version implementation is without multi-threading where pre-processors run sequentially. Our system gets input of an IFC model along with the rule set to be verified on the building model. It invokes conversion pre-processor, and loads the equivalent RDF triples into triple store. Then it runs the geometric pre-processor and gets the bounding box values (i.e., min/max values X, Y and Z coordinates) and updates the triple store with the geometric RDF triples. Finally Rule Evaluator runs the SPARQL queries to verify the compliance of building codes.

Table 2: Time taken by different Pre-Processors of SBIM-Reasoner

IFC model	IFC Size (Mb)	IFC-to-RDF Conversion (sec)	Geometry Extraction (sec)	Semantic Pre-Proc. (sec)	Rule Evaluator (sec)	Total (sec) version 1	Total (sec) Multi-Threading
HITOS	62.5	205	308	40	10	563	358
AC9R1-Haus	4.4	17	10	9	8	43	33
BIM_EM	1.9	16	10	7	7	40	30
Candidat-23_04	28.9	48	65	27	10	150	102
Chanteloup	17.1	41	31	18	10	100	69
LcD	32.4	97	103	31	9.5	241	144
Batiment_CSTB	14.9	40	27	11	9	87	60
Maq.Test Checker	11.2	32	19	8	8	67	48
HAixFlowCtrl	13.1	39	22	14	9	84	62
Liberty_V3	12.2	38	23	12	9	82	59

Later in our implementation which is the contribution of this paper, we adopted multi-threaded approach where geometric and conversion pre-processors run parallel to produce output RDF triples. In our test experiment, there are IFC models of various sizes and structures. An IFC model named HITOS is the largest building model which has 62.5 Mb size with 205,631 number of RDF triples and BIM_EM is the smallest building model which has 1.9 Mb size with only 2,138 number of RDF triples. When we see the time graph, we observe that *SBIM-Reasoner* took less than a minute by both the pre-processors 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 pre-processors took time to build semantic model at the first time, but querying for the verification of IFC models are processed fast and a good run-time 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. We revealed encouraging results via

several tests from the initial version of *SBIM-Reasoner*. The results of *SBIM-Reasoner* are checked manually to access the correctness of *SBIM-Reasoner* for each of the building models and they were 100% correct. On the basis of this analysis, we conclude that semantic model serves best for the verification of IFC models. We conclude that for such modules where execution can be done separately, multi-threading can benefit and gain time. It is highly visible when an IFC model is large (in the case of HITOS IFC model), parallel processing and CPU utilization reveals these pre-processors to execute simultaneously resulting the whole process take lesser time to achieve the whole functionality.

5. Conclusion

In the context of BIM when the question about the evaluation of an IFC model appears, the implementation of compliance checking of IFC models is vital to address. It is necessary to detection inconsistent and non-compliant IFC objects to ensure quality and reliability of an IFC model in the entire life-cycle of BIM. Compliance checking is an exhaustive process which cannot be done manually, in addition requires revolutionized advanced technology to process due to complex inherent nature of an IFC model itself. There are many techniques for the automatic verification of IFC models, but, still there are many open challenges. In this paper, we evaluated the idea to formulize building code rule as a semantic SPARQL query and then execute it on the building model to check the possibility of non-compliant elements in the building model. We have built a research prototype named *SBIM-Reasoner* which employs semantic approach for the building code conformance checking. We have tested and concluded that our approach based on semantic queries and rules can be easily extended, configured and deployed for the dynamic and changing BIM environment having broad spectrum of functionalities for the conformance checking of IFC models. We have also analyzed how multi-threaded approach can benefit our solution by comparing run-time of our two implementations with and without threading. 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.On the basis of several analysis parameters, we have shown encouraging results by several tests on the *SBIM-Reasoner*. We have also found that multi-threading based approach served best and provided faster computation for the conformance checking of IFC models.

References

Building Smart, International home of openBIM, https://www.buildingsmart.org/standards/ [last access: jan 2019]

Bouzidi, K. R., Fies, B., Faron-Zucker, C., Zarli, A. and Thanh N. Le. (2012), "Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry", Future Internet, Special Issue Semantic Interoperability and Knowledge Building, 4 (3). pp. 830-851.

Chipman, T., Liebich, T., and Weise, M. (2016) "mvdXML specification of a standardized format to define and exchange MVD with exchange requirements and validation Rules," version 1.1 Final, Febrary 2016. Available at: http://www.buildingsmart-tech.org/downloads/mvdxml /mvdxml-1.1/final/mvdxml-1-1-documentation

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

Fahad, M., Bus, N., Fies, B. (2014). “Semantic BIM Reasoner for the verification of IFC Models”, eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2017, Denmark

Friedman-Hill, E. (2003). “Jess in Action: Rule Based Systems in Java,” Manning Publications. ISBN 1-930110-89-8, 2003

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). “SWRL: A Semantic Web Rule Language Combining OWL and RuleML”.

IfcDoc Tool (2012), available at: http://www.buildingsmart-tech.org/specifications/specification-tools/IfcDoc-tool/IfcDoc-help-page-section/IfcDoc.pdf [last access: jan 2019]

Kadolsky, M., Baumgärtel, K., and Scherer, R.J. (2014). “An ontology framework for rule-based inspection of eeBIM-systems,” Procedia Engineering. vol. 85, pp. 293-301, doi:10.1016/j.proeng.2014.10.554.

Khemlani L. (2009). “Solibri model checker,” AECbytes Product Review March 31, 2009

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., and Hendler, J. (2008). "N3Logic: A logical framework for the World Wide Web," Theory and Practice of Logic Programming. vol. 8 (3), pp. 249-269, doi:10.1017/S1471068407003213.

Mendes de Farias, T., Roxin, A., and Nicolle, C. (2016) “A Semantic Web Approach for defining Building Views,” buildingSMART Summit Jeju, Korea.

Pauwels, P. and Oraskari, J. (2012) “IFC-to-RDF Converter” https://libraries.io/github/IDLabResearch/IFC-to-RDF-converter, [last access: jan 2019]

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.

Pauwels, P., Deursen, D. Van, Verstraeten, R., Roo, J. De, Meyer, R. De, Walle, R. Van de, and Campenhout, J. Van (2011). “A semantic rule checking environment for building performance checking,” Automation in Construction. 20 (5), pp. 506-518.

Rebekka, J.S., and Schultmann F. (2014). "Building Information Modeling (BIM) for existing buildings—Literature review and future needs," Automation in construction, vol. 38, pp. 109-127, 2014, doi: 10.1016/j.autcon.2013.10.023

SPARQL Query Language for RDF (2013), http://www.w3.org/TR/rdf-sparql-query/ [last access: jan 2019]

Thein, V. (2011). Industry Foundation Classes (IFC), “BIM Interoperability through a Vendor-Independent File Format,” A Bentley White Paper, September’11.

Terkaj, W., and Pauwels, P. (2014). “IfcOWL ontology file for IFC4”. Available at: http://linkedbuildingdata.net/resources/IFC4_ADD1.owl [last access: jan 2019]

Wicaksono, H., Dobreva, P., Häfner, P., and 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.

Zhang, C., Beetz, J. & Weise, M. (2014). “Model view checking: automated validation for IFC building models”. In Mahdavi, ed. eWork and eBusiness in Architecture, Engineering and Construction: ECPPM’14.