Semantic Rule-checking for Regulation Compliance Checking: An Overview of Strategies and Approaches

Pieter Pauwels, pipauwel.pauwels@ugent.be
Ghent University, Department of Architecture and Urban Planning, Belgium

Sijie Zhang, sijie.zhang@chevron.com
Chevron ETC, Houston, United States of America

Abstract
As more and more architectural design and construction data is represented in the Resource Description Framework (RDF) data model, it makes sense to take advantage of the logical basis of RDF and realise a semantic rule-checking process as it is currently not available in architectural design and construction industry. Such a semantic rule-checking process would be of considerable value to regulation compliance checking procedures, because the additional logical basis would (1) allow consistency checking of the rules in the regulations (and thus a better management of the rules), (2) allow a faster, more transparent and more reliable implementation of regulation compliance checking procedures (rules are human-readable and more easily changeable), and (3) provide the option to rely on available reasoning engines and generate proofs without any additional implementation effort. Obviously, there are a number of strategies and approaches that can be followed regarding the realisation of such a rule-checking process. In this article, we will outline three rule-checking approaches that have been reported for semantic rule-checking in the AEC domain. The produced comprehensive strategic overview can be used as a guide for parties interested in implementing a semantic rule-checking process for regulation compliance checking.

Keywords: rule checking, regulations, compliance, BIM, linked data, SWRL

1 Introduction

1.1 Types of rule-checking for construction industry

Rules and regulations are of key importance for the domain of architectural design and construction. In all sorts of forms and shapes, international, national, local and even personal rules need to be compliant with before, while and after a building is being built. Of particular importance are the national regulations that are applicable for specific buildings and constructions. The whole architectural design and construction industry is aware of the building energy performance regulations imposed by governments, but many other centrally imposed regulations exist as well, including acoustic performance regulations, safety regulations during construction phase, fire safety regulations, building access and exit regulations, and local urban regulations.

With the advent of Building Information Modelling (BIM) tools (Eastman et al, 2008), a central location has emerged in which building information can be managed at any point in time. Hence, automatic regulation compliance checking is within reach, as has also been acknowledge in detail by Eastman et al (2009). Automated rule checking is defined by Eastman et al (2009) as *"software that does not modify a building design, but rather assesses a design on the basis of the configuration of objects, their relations or attributes"*. Rule-based systems are hereby understood as systems that *"apply rules, constraints or conditions to a proposed design, with results such as 'pass', 'fail' or 'warning', or 'unknown'."* Many of the initiatives that started from a BIM model for regulation compliance checking started from a neutral representation of the building model, typically in the

Industry Foundation Classes (IFC). Eastman et al (2009) hereby refers to the early work by Han et al (1997, 1998, 1999).

Since more and more architectural design and construction data is now also represented in the Resource Description Framework (RDF) data model (Manola & Miller, 2004), the underlying logical basis of RDF can promote the realisation of the rule-checking process outlined by Eastman et al (2009) using this additional logical basis, as proposed earlier by Pauwels et al (2011). Also Kerrigan & Law (2003) indicated the usefulness of a logical basis in regulation compliance checking, as early as 2003, which was at that time implemented as an addition to plain XML. In this article, we will specifically look into the implementation methods that can be followed to allow the above outlined regulation compliance checking process. Eastman et al (2009) presents three different implementation methods, namely (1) computer language encoded rules, (2) parametric tables, and (3) language driven. Eastman et al (2009) further divides the language driven implementation method in two: either a predicate logic based language or a domain-oriented language is used. With its logical basis in Description Logics (DL) (Baader & Nutt, 2003), a semantic rule checking approach as proposed in Pauwels et al (2011) entirely fits in the former of the two (predicate logic based). The Built Environment Rule and Analysis (BERA) language (Lee, 2011; Lee et al, 2014) is an example that is closer to the latter of the two (domain-oriented language). Note, however, that semantic web languages (RDF and OWL) also allow to define domain oriented knowledge. Perhaps it is better to consider RDF as a predicate-logic based domain-oriented language. In Pauwels et al. (2011), a specific focus is set on the possibility of implementing the language driven implementation method, by using semantic web technologies (Berners-Lee et al, 2001), which is also the focus of this paper.

1.2 Example implementations of semantic rule-checking applications

A good number of applications have by now emerged that relies on the logical basis of semantic web languages to accommodate some form of semantic rule checking. One example that was already mentioned in the introduction is the effort by Pauwels et al (2011), which aims at accommodating acoustic regulation compliance checking for BIM models. In this exploratory article, an indication is made of the way in which N3Logic rules can be represented and used in combination with a domain ontology (TBox) and an instance model (ABox), so that an inference engine immediately indicates whether a building model is compliant or not with the European acoustic regulations. In a full implementation, one might opt to use the IFC data model to fill the TBox and ABox. A similar proposal is made in Pauwels et al (2011a) to convert a geometry representation in IFC to corresponding geometry representations in the X3D and STL schemas, thereby relying on N3Logic rules, the EYE reasoning engine and standard semantic web technologies.

Wicaksono et al (2010) rely on semantic web technologies to build an intelligent energy management system for buildings. They propose to build an RDF representation of a building model, using an OWL ontology for building information. This ontology includes concepts for the appliances present in the building (dishwasher, fridge). The OWL ontology can then be combined with a number of rules expressed in the Semantic Web Rule Language (SWRL), in order for an inference engine to infer if there are any anomalous activities occurring (e.g. 'heaters' that are 'working' AND 'windows' that are 'open'). This work has been extended in Wicaksono et al (2013), to include an OWL ontology inspired by IFC. In this extended proposal, the authors rely on a rule engine based on SWRLJessBridge, which allows the execution of rules combined with the Protégé API. A SPARQL endpoint is made available on top of this rule engine, so that the end user only has to query for the results of the rules (i.e. are the *EnergyInefficient* or *UsageAnomaly* individuals present?).

The third example showing the way in which rules can be deployed for construction industry and building information management is provided by Kadolsky et al (2014) and Baumgärtel et al (2015). In this example, the authors propose to represent a building in an ifcOWL ontology, after which rules can be used to retrieve information that is relevant for building energy performance. Baumgärtel et al (2015) specifically shows how rules can be used to allow a thermal insulation check: the right-hand side of one of the SWRL rules includes the statement *?summ eeBIM:definition "Thermal insulation check failed"*.

The last example is the Job Hazard Analysis (JHA) application that is documented by Zhang et al (2015). In this example, the authors propose to combine an RDF representation of the building

model, contained in Tekla Structures, with a number of ontologies and SWRL rules that allow the analysis of the construction project in terms of jobs, tasks, safety procedures, and the resources that are required to allow the safe execution of these job steps. A prototype was implemented in the form of a plugin in Tekla Structures.

As these examples successfully illustrate, the usage of semantic web technologies in Architecture, Engineering and Construction (AEC) industries opens up considerable possibilities in terms of formal rule-checking. This rule-checking can be useful for a number of different use cases, including building usage analysis, anomaly detection, job hazard analysis and regulation compliance checking. In the following section, we will recapitulate the basics of semantic rule checking, thereby outlining some of the key principles. Thereafter, we will look more closely in the implementation strategies that can be followed to accommodate semantic rule-checking.

2 The basics of semantic rule checking

At the core of the regulation compliance checking process are rules that need to be checked. To allow proper checking of rules, three components are always necessary: (1) a schema that defines what kind of information is used by the rule checking process and how it is structured, (2) a set of instances following that schema, and (3) a set of rules (IF-THEN statements) that can be directly combined with the schema (the rule set contains the classes and properties that are defined in the schema). These three components are filled in in various ways. In a traditional hard-coded rule checking process, the schema is typically represented by the class hierarchy of the coded system, the instances are represented by the objects that follow this class hierarchy, and the rules are represented by procedural interconnected functions that can follow any kind of structure, while still being compatible with the class hierarchy of the system. This is considerably different from the way in which these three components take shape in a semantic, language driven approach. Namely, in this case, the schema is typically represented by an OWL ontology (McGuinness & van Harmelen, 2004), the instances are represented by the RDF graph following that OWL ontology, and the rules are logical conjunctions (AND) of declarative IF-THEN statements. Because of the logical basis of the OWL language in DL (Baader & Nutt, 2003), the rule checking process is quite straightforward as soon as all the data and all the rules are available in a complete and consistent shape: one generates the inferences and uses the results, e.g. for simple visualisation in a graphical user interface (GUI).

The great advantage of using semantic web technologies is that the schema, the instances, and the rules can all be described using one and the same language. As a result, all three components benefit from the advantages given by Eastman et al (2009) for any language-driven approach, namely, (1) the possibility to easily retarget an implementation to different source formats (e.g. an alternative ontology: a Revit ontology instead of an IFC ontology); (2) portability across contexts, applications and devices, and (3) the availability of an unlimited representation wealth, including 'nested conditions' and 'branching of alternative contexts'. The downside of this feature is that data, ontologies and rules can be stored in very diverse ways and environments. For example, in some cases, they are shared openly on the World Wide Web (WWW); in other cases, only the ontologies are widely available via the WWW and the rules and the data are kept in local applications; in other cases, all three components are kept solely as in-memory models of an application, making them accessible only via the programming code; and it can even be imagined that all three components are generated on-demand from legacy data sources (e.g. SQL databases that are given an RDF interface using R2RML (Das et al, 2012)). This results in a number of implementation plans, each of them having its strengths, weaknesses, opportunities, and strengths.

In the next sections, we will go through the key strategies found in the literature and briefly outline our experiences with them for a small use case extracted from the "Job Hazard Analysis" checking system documented in Zhang et al (2015). We will specifically use one rule that is used in this use case (see below), and document how it can be stored and used in the diverse implementation plans.

```
  1 Masonry_Wall(?mw) ∧ hasHeight(?mw, ?h) ∧ Task_Masonry_Wall(?act)
∧produce(?act,
  2 ?mw) ∧ consistOf(?act, ?sub) ∧ Masonry_Operation(?sub) ∧
consistOf(?sub, ?pb) ∧
```

```
  3 Placing_Brick(?pb) ∧ swrlb:greaterThan(?h, 2438.4) →
needResources(?pb,
  4 Masonry_Wall_Bracing)
```

This rule allows to infer that a 'masonry wall' needs to be 'braced' when it is 'produced' by 'placing bricks' and the 'height' of this wall is greater than '2438,4'. As it is displayed above, it is not contained yet in any kind of software. There are a number of options to formally represent it in a software environment, depending on which kind of implementation is chosen. These options are discussed in the following three sections, thereby indicating what they result in for the usability of these rules by developers and end users.

3 Strategy 1: Hard-coded rule checking after querying for information

The first strategy most closely ties to existing procedures for rule-checking and existing software implementation plans. Namely, it is possible to represent the information that is contained in the rule mentioned earlier as plain RDF data, ideally following an OWL ontology. Many existing applications for rule-checking take this approach, although they usually do not use RDF and OWL to store the rule information. For example, the commercial Solibri Model Checker application (Solibri, 2014) provides a great interface that allows to load a BIM model and combine it with information that is stored natively in the Solibri application (database). This then allows performing rule checking against BIMs for architectural design validations. In most of the existing applications, including Solibri, the rules are not available outside the actual application. As a result, the user is required to manage and use rules through that interface. When relying on RDF and OWL, this can be done differently. For example, in the below code, a representation is given of the rule that was displayed earlier.

```
  1    ruleInst:wallBracingRule
  2      rdf:type ruleOnt:Rule ;
  3      ruleOnt:inputInformation "a masonry wall with a height greater
than 2438,4, built
  4        by placing bricks" ;
  5      ruleOnt:outputInformtion "bracing of masonry wall required" ;
  6      ruleOnt:partOfRegulation ruleInst:jobHazardAnalysis_7390 .
```

If the above rule information, which is represented in RDF, is published in an open or somehow accessible repository, for instance a government-owned repository, anyone can access this information and use it. Access of the information can occur via a visual web interface (e.g. Pubby), via a SPARQL query endpoint, via an API, or via any other means that is provided by the owner of the RDF data. This is different from what currently happens in applications like Solibri, in the sense that the rules are external to the application. An example application of this strategy is provided and documented by Dimyadi et al (2014, 2015) for the New Zealand Building Code (NZBC) for fire safety design of buildings. This building code is available from the New Zealand government in plain text. A formal representation of this code is prepared, both in XML and RDF. An application has then be developed that queries this formal representation and makes actions accordingly. A similar approach might be followed by applications like Solibri, simply if they allow to load rules not only from an internal database, but also from external formal representations (in XML or RDF) of rule information.

It is important to note here that the success of the rule-checking process still considerably depends on the algorithms of the rule-checking application itself. Namely, the application needs to find out when to send which kinds of queries to the XML and/or RDF representations, and it needs to consume the returned information into executable code that actually performs the rule-checking. For example, in the case of an application like Solibri, application developers will likely opt to directly execute rule-checking processes on BIM models and checking results can then be visualised directly in a 3D representation of the BIM model. Other software developers, depending on their preferences or business model, can accommodate an entirely different procedure. The formal representation of the rules is only a small part of the success of the rule-checking procedure. Second, and perhaps more importantly, the source code of the application needs to match with the

formal representation of the rules and 'understand' how to interpret the rule information it retrieves from the external dataset. As a result, there is only limited support possible for rule customisation. If the rules are customised beyond what can be interpreted by the source code of the rule-checking application, the application can no longer use those rules. Also, its rule checking capability is restricted to check whether information exists and is available in the model. In other words, the rule checking cannot be achieved if the required information does not exist or is not accessible in the building information model.

4 Strategy 2: Rule-checking by querying

An alternative approach has been proposed by Yurchyshyna et al (2008, 2008a), Yurchyshyna (2010), Bouzidi et al (2011, 2012). In their proposal, the rule information is formalised directly into SPARQL queries. This is different from the previous approach, in the sense that only the building model is represented in RDF, but not the rule information. This building model is then 'interrogated' in detail using formal SPARQL queries. The rule information is captured in these SPARQL queries. The example below shows what this results in for our example presented earlier.

```
1    SELECT *
2    WHERE {
3       ?mw rdf:type ont:Masonry_Wall .
4       ?mw ont:hasHeight ?h .
5       ?act ont:produce ?mw .
6       ?act rdf:type ont:Task_Masonry_Wall .
7       ?act ont:consistOf ?sub .
8       ?sub rdf:type ont:Masonry_Operation .
9       ?sub ont:consistOf ?pb .
10      ?pb rdf:type ont:Placing_Brick
11      FILTER (?h >= 2438,4)
12   }
```

The example code above shows a simple query. The results of this query are then to be interpreted by the rule-checking application in order to visualise the end result and inform the end user. This procedure is somewhat comparable to the previous approach. The main difference here is that the actual rule information is easier to represent using SPARQL queries instead of using the flat RDF data model, which is more appropriate for representing the actual data. Furthermore, when relying on the CONSTRUCT keyword available in SPARQL, an even closer formalisation of the rule information lies within reach (see example below).

```
1    CONSTRUCT { ?pb ont:needResources [ rdf:type
ont:Masonry_Wall_Bracing ] }
2    WHERE {
3       ?mw rdf:type ont:Masonry_Wall .
4       ?mw ont:hasHeight ?h .
5       ?act ont:produce ?mw .
6       ?act rdf:type ont:Task_Masonry_Wall .
7       ?act ont:consistOf ?sub .
8       ?sub rdf:type ont:Masonry_Operation .
9       ?sub ont:consistOf ?pb .
10      ?pb rdf:type ont:Placing_Brick
11      FILTER (?h >= 2438,4)
12   }
```

By executing this query for a building model, the building model will generate the additional *"needResources"* property when the query is successful. This property can be added to the actual building model, thus enriching the data model, or it could be kept separate for processing by alternative algorithms. If a number of such queries are executed, one might be able to assess step by step the conformance of a building with certain rules or requirements. This procedure can be further extended by relying on the SPARQL Inference Notation (SPIN). As indicated in SPIN (2015),

this method relies heavily on the CONSTRUCT keyword in SPARQL and the SPARQL INSERT and UPDATE statements. The downside of the strategy of implementing rule-checking by querying is that queries typically still need to be fired one by one by an application logic. This thus requires some implementation work. Obviously, the SPARQL queries also need to be in-sync with the ontologies used to express the building information.

5 Strategy 3: Semantic rule checking with dedicated rule languages

5.1 Theory and formal notation

The third and last approach relies on dedicated semantic web rule languages. Two example rule languages are SWRL and N3Logic. SWRL is one of the most often used languages (Horrocks et al, 2004). Other languages are Jess (Friedman-Hill, 2008) and N3Logic (Berners-Lee et al, 2008; Vanel, 2012). Rules can be exchanged using the Rule Interchange Format (RIF, 2013). Several rule engines are able to process the rules in combination with RDF data and OWL ontologies. The Jess Rule Engine (Friedman-Hill, 2008) is entirely built in Java, so this can be combined easily with native Java applications and Jess rules. Similarly, the Apache Jena software library was implemented in Java, which also includes Jena rules and functionality to execute these rules with RDF data and OWL ontologies opened up in-memory in the Jena-enabled application (Apache Jena, 2015). N3Logic rules are easily consumed by the Euler Yap Engine (De Roo, 2011, 2015). SWRL rules are often used within Protégé. A SWRLAPI is also available that allows to build, process and execute rules in a software application (O'Connor et al, 2005; O'Connor & Dias, 2012).

In construction industry, this third strategy has been used most often in recent research for both compliance checking and performance control. Literature examples mentioned in the introduction of this article show its application to assist construction quality compliance checking in Zhong et al (2012), to monitor and control energy consumption in Wicaksono et al (2010), to enable BIM-based job hazard analysis in Zhang et al (2015), and to support the plan definition and verification process in pit excavation in Zhong et al (2015). Depending on the scenario, the respective authors decide to choose one or the other rule language and inference engine. The way in which the rule information is represented using any one of the rule languages is very similar to the original form of this rule information. Namely, rule languages allow to formally specify RDF graphs, both in the left hand side and in the right hand side of the formal notation, connecting them with a logical *"implies"*. We provide an example below in the N3 Logic notation, which can easily be transformed into equivalent notations in the other rule languages (Jess, SWRL, and others).

```
{
  ?mw rdf:type ont:Masonry_Wall .
  ?mw ont:hasHeight ?h .
  ?act ont:produce ?mw .
  ?act rdf:type ont:Task_Masonry_Wall .
  ?act ont:consistOf ?sub .
  ?sub rdf:type ont:Masonry_Operation .
  ?sub ont:consistOf ?pb .
  ?pb rdf:type ont:Placing_Brick .
  ?h math:greaterThan 2438,4
}
log:implies
{
  ?pb ont:needResources [ rdf:type ont:Masonry_Wall_Bracing ]
}
```

This rule is in fact following a triple notation, with the left hand side of the rule being the 'subject' of the statement, the *log:implies* property as the 'predicate' of the statement, and the right hand side of the rule being the 'object' of the statement. So, this is not only a very natural way of describing any kind of rule information (RBox), it is also highly compatible with OWL ontologies (TBox) and RDF instance graphs (ABox). In this strategy, an application targeting at rule-checking procedures thus typically aims at storing three things separate from each other: TBox, ABox and RBox. On top of these three, there is only a reasoning engine to combine pieces from the TBox,

ABox and RBox, and an application logic that allows to visualise the inferred information in one way or another (Figure 1).

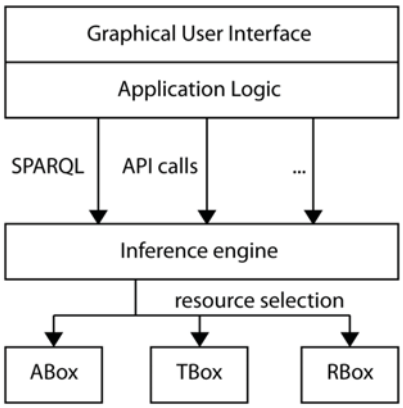

Figure 1. Overview implementation path for any rule-checking application relying on a dedicated rule language.

For each of the components, a different library can be used (e.g. SWRL instead of N3Logic for representing the RBox; the Jess rule engine instead of the SWRLAPI; standalone application instead of plugins in existing software, SPARQL versus API calls, ...). But at the core, the overall implementation path remains the same.

5.2 Application in construction industry

A use case on BIM-based hazard identification and mitigation from Zhang et al (2015) shows detailed ontology development, instance generation, SWRL rules checking using rule engine, and finally its checking results visualisation back in BIM. It first formalises construction safety knowledge using engineering ontology. Then, selected OSHA regulations and industry safety best practices are coded in SWRL rule formats, compatible with ontology classes and relationships. The used SWRL rule is the masonry wall example shown in Section 2. A BIM-based JHA application is developed to automatically identify work activity related safety hazards, suggest mitigation methods, and visualise relevant safety information, such as hazard zones. The system architecture of the ontology-based hazard identification application includes ontology editor, reasoner, rule engine, and BIM platform (Figure 2).

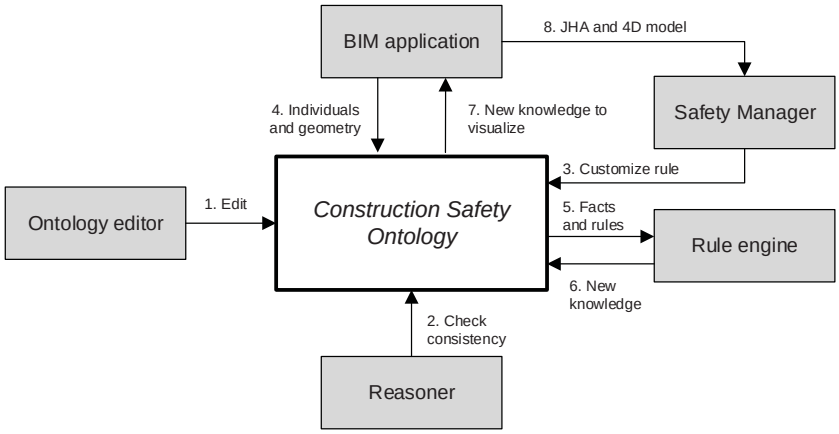

Figure 2. System architecture of the ontology-based hazard identification application in BIM.

1) The OWL-based safety ontology is first modelled and edited using an ontology editor, in this case Protégé, to define its classes, relationships and axioms.

2) The consistency of the developed Construction Safety Ontology is then checked by OWL reasoner (Pellet).

3) Based on the Construction Safety Ontology, SWRL rules are then developed to represent OSHA regulations and industry best practices. Also, the rule set can be customised and rules can be added by subject matter experts according to their specific requirements.

4) After connecting the ontology with BIM software platform, individuals/instances of the safety concepts defined in the ontology are generated using BIM project information. Properties of each individual, such as geometry information, are obtained through BIM.

5) Facts including the knowledge base and individuals generated from BIM are passed to the Jess rule engine to be checked against SWRL rules defined earlier by a safety manager or safety superintendent.

6) Once new knowledge is inferred by the rule-checking process, the ontology is updated by the rule engine.

7) The updated OWL ontology is then linked with the BIM platform to visualise inferred knowledge, such as required safety protective systems and protective safety zones.

8) Finally, project specific JHA along with a 4D building model visualising safety information are generated to support site level project safety planning and inspection.

6 Conclusion: comparison across rule-checking strategies

We presented a non-exhaustive overview of key implementation strategies for semantic rule-checking tailored to the architectural design and construction industry. A comparison between the three rule-checking techniques is shown in Table 1.

Table 1 Comparison between the three rule-checking techniques

	Coded rule-checking	Rule-checking by querying	Dedicated rule language
Time	Short	Long	Long
Customization	Limited	Open	Open
Knowledge inference	No	No / Limited	Yes
Integration with BIM	Direct or Indirect	Direct or Indirect	Direct or Indirect

(1) The time requirement of coded rule-checking is very short since the checking algorithms can be executed directly on the model. Rule-checking by querying and rule-checking with rule languages takes longer, as they require processing of queries / rules in combination with a model that is often too large. When implementing an approach that relies on a dedicated rule language, it is of the utmost importance to be as selective as possible at the 'resource selection' stage in Figure 1.

(2) The user rule-customisation is limited for coded rule-checking since a user interface needs to be provided for such customisation. SPARQL queries and SWRL-like rules offer more flexibility. The user can define new SPARQL queries or SWRL-like rules to check against new or customised conditions, as long as the information exists in the ABox and TBox (Figure 1).

(3) Knowledge inference is only supported by the approach based on a rule language. This is a key methodological difference compared to the other two approaches (declarative versus procedural).

(4) Depending on the way in which the application logic and GUI is implemented (cfr. plug-in in existing BIM software or standalone applications), rule-checking can be directly (tightly) or indirectly (loosely) combined with available BIM software. In any case, additional programming is required in all three approaches.

Acknowledgements

This research is made possible by the generous support of Ghent University, through the Special Research Fund (BOF).

References

Apache Jena (2015). Reasoners and rule engines: Jena inference support. https://jena.apache.org/documentation/inference/ (Last visited on 12 June 2015).

Baader, F. & Nutt, W. (2003). Basic description logics. In: *Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge. pp. 47-100.

Baumgärtel, K., Kadolsky, M. & Scherer, R.J. (2015). An ontology framework for improving building energy performance by utilizing energy saving regulations. *Proc. of the 10th European Conference on Product & Process Modelling (ECPPM)*. pp. 519-526.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web. *Scientific American*. 284 (5). pp. 29-37.

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y. & Hendler, J. (2008). N3logic: A logical framework for the world wide web. *Theory and Practice of Logic Programming*. 8 (3). pp. 249-269.

Bouzidi, K.R., Fies, B., Bourdeau, M., Faron-Zucker, C. & Le Thanh, N. (2011). An ontology for modelling and supporting the process of authoring technical assessments. *Proc. of the CIB W78-W102 2011 International Conference*.

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

Das, S., Sundara, S. & Cyganiak, R. (2012). R2RML: RDB to RDF Mapping Language. W3C Recommendation 27 September 2012. http://www.w3.org/TR/r2rml/ (Last visited on 12 June 2015).

De Roo, J. (2011). Euler YAP Engine, a birds EYE view. http://www.agfa.com/w3c/Talks/2011/01swig/ (Last visited on 12 June 2015).

De Roo, J. (2015). Euler Yet another proof Engine. http://eulersharp.sourceforge.net/ (Last visited on 12 June 2015).

Dimyadi, J. & Amor, R. (2013). Regulatory knowledge representation for automated compliance audit of BIM-based models. *Proc. of the 30th CIB W78 International Conference*. pp. 68-78.

Dimyadi, J., Pauwels, P., Spearpoint, M., Clifton, C. & Amor, R. (2015). Querying a Regulatory Model for Compliant Building Design Audit. *Proc. of the 31st CIB W78 International Conference*.

Eastman, C., Teicholz, P., Sacks, R. & Liston, K. (2008). *BIM Handbook: A guide to Building Information Modeling for Owners, Managers, Architects, Engineers, Contractors, and Fabricators*. John Wiley and Sons, Hoboken, NJ.

Eastman, C., Lee, J.-M., Jeong, Y.-S. & Lee, J.-K. (2009). Automatic rule-based checking of building designs. *Automation in Construction*. 18. pp. 1011-1033.

Friedman-Hill, E. (2008). Jess, the rule engine for the Java Platform. http://www.jessrules.com/ (Last visited on 12 June 2015).

Han, C., Kunz, J. & Law, K. (1997) Making automated building code checking a reality. *Facility Management Journal*. pp. 22-28.

Han, C., Kunz, J. & Law, K. (1998) Client/server framework for on-line building code checking. *ASCE Journal on Computing in Civil Engineering*. 12 (4). pp. 181-194.

Han, C., Kunz, J. & Law, K. (1999) Building design services in a distributed architecture, *ASCE Journal on Computing in Civil Engineering*. 13 (1). pp. 12-22.

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. W3C Member Submission 21 May 2004. http://www.w3.org/Submission/SWRL/ (Last visited on 12 June 2015).

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

Kerrigan, S. & Law, K. (2003). Logic-based regulation compliance-assistance. *Proc. of the Ninth International Conference on Artificial Intelligence and Law*.

Lee, J.K., Eastman, C.M. & Lee, Y.C. (2014). Implementation of a BIM Domain-specific Language for the Building Environment Rule and Analysis. *Journal of Intelligent & Robotic Systems*. pp. 1-16.

Lee, J.K. (2011). Building Environment Rule and Analysis (BERA) Language and its application for evaluating building circulation and spatial program. PhD thesis. Georgia Institute of Technology.

Manola, F. & Miller, E. (2004). RDF Primer, W3C Recommendation. http://www.w3.org/TR/rdf-primer/ (Last visited on 12 June 2015).

McGuinness, D.L. & van Harmelen, F. (2004). OWL Web Ontology Language Overview - W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-features/ (Last visited on 12 June 2015).

O'Connor, M.J., Knublauch, H., Tu, S.W., Grossof, B., Dean, M., Grosso, W.E. & Musen, M.A. (2005). Supporting Rule System Interoperability on the Semantic Web with SWRL. *The Semantic Web – ISWC 2005*. In: *Lecture Notes in Computer Science*. 3729. pp. 974-986.

O'Connor, M.J. & Das, A. (2012). A Pair of OWL 2 RL Reasoners. *9th International Workshop on OWL: Experiences and Directions (OWLED)*.

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

Pauwels, P., Van Deursen, D., De Roo, J., Van Ackere, T., De Meyer, R., Van de Walle, R. & Van Campenhout, J. (2011a). 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). pp. 317-332.

RIF (2013). Rule Interchange Format Current Status. http://www.w3.org/standards/techs/rif#w3c_all (Last visited on 12 June 2015).

Solibri (2014). http://www.solibri.com/solibri-model-checker.html (Last visited on 12 June 2015).

SPIN (2015). SPIN SPARQL Inferencing Notation. http://spinrdf.org/ (Last visited on 12 June 2015).

Vanel, J.M. (2012). Logic and rules - N3 Logic and syntax. http://eulergui.sourceforge.net/documentation.html#Logic (Last visited on 12 June 2015).

Wicaksono, H., Rogalski, S. & Kusnady, E. (2010). Knowledge-based intelligent energy management using building automation system. *Proc. of the 2010 IPEC Conference*. pp. 1140-1145.

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 International Conference on Knowledge Engineering and Ontology Development*.

Yurchyshyna, A., Faron-Zucker, C., Le Thanh, N. & Zarli, A. (2008). Towards an ontology-based approach for formalising expert knowledge in the conformity-checking model in construction. *Proc. of the 7th European Conference on Product and Process Modelling (ECPPM).*

Yurchyshyna, A., Faron-Zucker, C., Le Thanh, N., & Zarli A. (2008a). Towards an Ontology-enabled Approach for Modeling the Process of Conformity Checking in Construction. *Proc. of the CAiSE'08 Forum*, pp. 21-24.

Yurchyshyna, A. (2010). Knowledge capitalisation and organisation for conformance checking model in construction. *International Journal of Knowledge Engineering and Soft Data Paradigms*. 2 (1). pp. 15-32.

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

Zhang, S., Teizer, J., Lee, J.K., Eastman, C.M. & Venugopal, M. (2013). Building information modeling (BIM) and safety: Automatic safety checking of construction models and schedules. *Automation in Construction*. 29. pp. 183-195.

Zhong, B.T., Ding, L.Y., Luo, H.B., Zhou, Y., Hu, Y.Z. & Hu, H.M. (2012). Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking. *Automation in Construction*. 28. pp. 58-70.