www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

MODELLING AND ACCESSING REGULATORY KNOWLEDGE FOR COMPUTER-ASSISTED COMPLIANCE AUDIT

SUBMITTED: July 2016
REVISED: September 2016
PUBLISHED: November 2016 at http://www.itcon.org/2016/21
GUEST EDITORS: Dimyadi J. & Solihin W.

Johannes Dimyadi, Postdoctoral Researcher,
Department of Computer Science, University of Auckland, New Zealand;
j.dimyadi@auckland.ac.nz

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

Robert Amor, Professor,
Department of Computer Science, University of Auckland, New Zealand;
trebor@cs.auckland.ac.nz

SUMMARY*: The ingredients for an effective automated audit of a building design include a building model containing the design information, a computerised regulatory knowledge model, and a practical method of processing these computable representations. There have been numerous approaches to computer-aided compliance audit in the AEC/FM domain over the last four decades, but none has yet evolved into a practical solution. One reason is that they have all been isolated attempts that lack any form of industry-wide standardisation. The current research project, therefore, focuses on investigating the use of the industry standard building information model and the adoption of open standard legal knowledge interchange and executable workflow models for automating conventional compliant design processes. This paper provides a non-exhaustive overview of common approaches to model and access regulatory knowledge for a compliance audit. The strengths and weaknesses of two comparative open standard knowledge representation approaches are discussed using an example regulatory document.*

KEYWORDS*: Knowledge management, regulatory compliance audit, compliant design workflows, regulatory knowledge, domain specific language.*

REFERENCE: *Johannes Dimyadi, Pieter Pauwels, Robert Amor (2016). Modelling and accessing regulatory knowledge for computer-assisted compliance audit. Journal of Information Technology in Construction (ITcon), Special issue: CIB W78 2015 Special track on Compliance Checking, Vol. 21, pg. 317-336, http://www.itcon.org/2016/21*

1. INTRODUCTION

Research in the area of computer-aided compliant design audit processes in the Architecture, Engineering, Construction and Facility Management (AEC/FM) domain dates back to the 1960s when decision tables were first utilised to aid engineering design for conformance with the AISC (American Institute of Steel Construction) specifications (Fenves, Gaylord, & Goel, 1969). This led to the development of expert systems such as SASE, SICAD, SPEX (Fenves & Garrett, 1986), but none of them survived, partially as the result of high maintenance costs in keeping up with frequent changes in the expert knowledge. Since then, there has been a new stream of projects and prototype systems being developed in Australia, New Zealand, Singapore, Finland, Sweden, Norway, Germany, France and the USA, adopting a variety of approaches including the widely popular rule-based systems. Other suggested approaches include the use of hypertext and hypermedia to aid navigating regulatory texts, automated and semi-automated knowledge acquisition from regulatory texts by means of deontic modelling, natural language processing (NLP), document markup techniques, and the use of a domain ontology and semantic technologies with reasoning capabilities (Evt, Khayyal, & Sanvido, 1992; Pauwels et al., 2011; Salama & El-Gohary, 2013; Zhang & El-Gohary, 2013; Zhong et al., 2012).

An effective regulatory knowledge representation must allow correct and efficient on-demand access to information. This paper reviews some commonly used methods of regulatory information retrieval and describes the capabilities of using two practical approaches to aid performance-based compliant design processes, namely one based on LegalRuleML (Athan et al., 2013; Palmirani et al, 2011) and one based on rule languages in the semantic web (Berners-Lee et al., 2001).

The paper is an extension of the work presented at the 32nd International CIB W78 Conference in Eindhoven, the Netherlands, in October 2015 (Dimyadi et al., 2015). We will first consider diverse existing implementation approaches towards the representation and access of regulatory knowledge for compliance audits in the AEC/FM domain (Section 2). We particularly focus on the key outstanding challenges regarding modelling and accessing regulatory information. Section 3 briefly outlines the regulatory document that we are using for testing purposes in this paper, namely the C/VM2 compliance document that is part of the New Zealand Building Code (NZBC). Sections 4 and 5 then briefly outline how requirements in this document can be encoded using the LegalRuleML and the semantic web approach, respectively. Section 6 finally discusses how both presented approaches can be related and perhaps complement or strengthen each other with further investigations.

2. REPRESENTING AND ACCESSING REGULATORY KNOWLEDGE FOR COMPLIANCE AUDIT

2.1 Key Challenges in Representing and Accessing Regulatory Knowledge

Regulatory knowledge in the context of the AEC/FM domain includes a lot of explicit forms of knowledge, such as prescribed design parameters, mathematical equations, rules, constraints and other normative data. This knowledge is conventionally written in natural language texts for human interpretation, yet there are many ways in which a design can be made compliant with these explicit regulations. A designer may choose certain parameters and scenarios to achieve a particular compliant design. We refer to this sequence of choices as a 'path to compliance'. A considerable number of such paths exist in legal documents and standards used for design. If a different set of parameters or scenarios were chosen, then another path may be found to achieve a new compliant design solution. Ultimately, it is up to the designer to evaluate and decide which path to follow. The decision on which compliance path to take often depends on the implicit knowledge that takes into account selected design scenarios, acceptable levels of risk, and safety margins.

As all natural language text is subject to human interpretation, it is always ambiguous and can thus entail many different meanings depending on the reader. Representing these texts in the unambiguous and explicit representations necessary for computers to process requires a person to select one unambiguous and explicit representation of the regulatory knowledge that is present in the text. Ideally, this one selected representation is either as widely applicable as possible (if the unambiguous representation serves a generic purpose), or it matches as closely as possible with the considered application case (if the unambiguous representation serves a single relatively well-defined purpose).

The implicit regulatory knowledge also considers selected building performance criteria, which are usually descriptive and not prescriptive in nature. Contemporary building design solutions are driven by high-performance objectives and innovation, which often fall outside the scope of prescriptive regulatory requirements. Performance criteria usually require evaluations by means of engineering analysis or simulations, which are not easily represented in a rule-based system. Performance-based regulations allow designers to explore engineering solutions from a broad range of compliant design options, which means that the number of compliance paths is often indeterminate at the outset.

We can thus outline three key challenges in representing and accessing regulatory knowledge:

1. Multiple paths to compliance:
   There are many parameters and scenarios that affect the way in which a design can be made compliant with these explicit regulations.
2. Ambiguity in regulatory documents:
   A regulatory text can entail many diverse meanings depending on the (human) interpreter, whereas the digital representation can only capture one unambiguous meaning.
3. Implicit regulatory knowledge:
   Certain implicit regulatory knowledge, such as performance criteria that require analyses or simulations, is hard to formalise.

So, while the more explicit regulatory knowledge can generally be formalised relatively easily into rules, implicit knowledge (compliance paths, ambiguity, performance criteria) is much more difficult to represent. There needs to be a more practical method of representing and accessing such implicit regulatory knowledge so that it supports human input and allows interactions with engineering analysis or simulation tools commonly used by building designers.

In this research we, therefore, argue that building designers should:

1. accept the responsibility of specifying exactly which objects or attributes in a regulatory model and a building model are to be checked for compliance.
2. specify one or more mapping tables between building objects and objects used in regulation texts (e.g. walls versus space boundaries), so that a compliance audit workflow is available for an automated design compliance audit with multiple iterations.
3. specify the input and output schemas of engineering analysis and simulation tools so that the design compliance audit tool can use this mapping and the required information can be automatically obtained from these tools if required.

Of course, designers should first have the tools and technologies that allow them to do these tasks. A number of tools have already been proposed in the past. Sections 2.2 and 2.3 outline two main approaches that are typically followed in the implementation of these tools.

2.2 Conventional Hard-coding Approach

When aiming at compliance audits, many of the conventional systems follow a hard-coding implementation approach. This means that most regulatory knowledge is embedded in the software code and is thus unreachable for anyone but a systems programmer, although some allow limited access and modification of their rules through predefined parameters (e.g. 'parametric tables' in Eastman et al. (2009)). The code typically follows a rule-based structure and approach, in the reflection of the rule-based structure of regulatory knowledge.

A common method of developing a rule-based system is to manually extract and translate written rules directly into computer code, optionally using parameterisation and branching. In this approach, formalised regulatory information in the form of codified rules is then accessed internally by the programming code of the compliance audit application. A comprehensive survey of conventional rule-based compliance audit tools and prototype systems has been given in the literature (Eastman et al., 2009). Hard-coded rules are indicated to be central in these conventional systems, resulting in rules that are tightly integrated into the compliance audit system, e.g. DesignCheck, SMARTCodes, ePlanCheck, Solibri Model Checker (Eastman et al., 2009). One challenge associated with hard-coded rules as part of the compliance audit system is the inflexibility and high cost to update, as it requires a system programmer to recode the system to accommodate even a minor rule change.

Furthermore, a proprietary or closed rule-based system lacks the transparency that makes it possible for end users or domain experts to verify the correctness of the implementation. The approach in which rules are hard-coded in the software may be acceptable for representing prescriptive regulatory requirements in specific applications, but it is far from adequate for representing performance-based requirements due to their dynamics and qualitative or descriptive nature.

Key decisions in implementing hard-coded compliance audit tools are typically characterised by the way in which one handles object mapping and separation of rules from the code.

2.2.1 Object Mapping

Systems that are designed to automatically compare objects in multiple data representations (e.g. building model and regulation text; or CSV and XML) typically need to rely on object mapping schemas between those data representations. As a simple example, in order to check if a doorway in a building model has an adequate width for compliance with certain regulations, the equivalent doorway object in the relevant regulatory model needs to be first identified and the required width attribute in a given condition noted. An automated compliance audit system has a challenging task to check every object and its attributes in the building model independently against the equivalent objects and attributes in the regulatory model, taking into account any condition or scenario attached to the use of those objects.

In a hard-coded compliance audit system, the object mapping is typically achieved by loading all different data representations into a central database and performing complete 1-to-1 mappings between pairs of representations. The mapping is typically an inherent and crucial part of the compliance audit system. The internal mapping process typically relies on a number of mapping files, which indicate which objects can be considered identical within pairs of data representations. Attributes can then be merged and/or compared for compliance audits.

2.2.2 Separating rules from the code

Separating rules from the compliance audit core functionality improves maintainability, enables extensibility, and allows portability. If rules are separated, regulatory requirements are formalised into a set of IF-THEN statements and stored in a centralised database that is accessible by a rule engine. Rule engine implementations usually allow selecting, chaining and executing one or more rules as required in a runtime production environment.

There are many open standard rule engines that may be suitable to represent legal knowledge relevant to the AEC/FM domain for compliance audit purposes, for example, DROOLS, OpenRules and OpenRuleEngine, SRE (Simple Rule Engine), JESS (Java Expert System Shell), and others. In particular, DROOLS and its DROOLS Rule Language (DRL) has been suggested as a feasible method for representing regulatory knowledge in a number of research projects on computer-aided compliance audit for the domain (Beach et al, 2013; Solihin & Eastman, 2015). A DRL rule consists of:

- a name;
- a number of attributes, which indicates the character of the rule and how it should be parsed or used;
- a left-hand side (LHS), which contains the conditional IF statements of the rule;
- a right-hand side (RHS), which contains the resulting THEN statements of the rule.

An example DRL rule is given in Listing 1.

Listing 1: An example DRL rule

```
rule "Hello World"
      dialect "mvel"
  when
      m : Message(status == Message.HELLO, message : message)
  then
      System.out.println(message);
      modify (m) {message = "Goodbye cruel world", status = Message.GOODBYE};
end
```

As can be seen from this example, these rules can include many procedural statements. They are hence not that different from a hard-coded approach. Yet, they do provide the great advantage that regulatory knowledge can be split out of the entire code of a compliance audit system, resulting in a compliance audit system that has access to somewhat independent modular rule sets. This feature results in the advantages of a rule-based system as it was outlined by (Eastman et al., 2009), namely, rule portability and flexibility.

2.3 Language-based compliance audit approach

An alternative to the conventional hard-coding approach is to represent regulatory knowledge using a dedicated rule language instead of computer code. Eastman et al., 2009 distinguish between the usage of a domain-specific language and a language that is based on a particular logic. To some extent, this language-based approach extends the DROOLS approach as briefly mentioned above, in the sense that regulatory knowledge would be available as modular rule sets (flexibility and portability). In addition, however, the regulatory knowledge would be represented using a language with a specific formal basis (the declarative approach in Pauwels et al. (2011)), instead of the computer code that can follow any number of incompatible ratios at the same time in the worst case scenario (the procedural approach in Pauwels et al. (2011)). Such an approach would thus make regulatory knowledge and compliance audits even more flexible, transparent, and portable.

Note that these language-based compliance audit tools are also characterised by key decisions regarding object mapping and separation of rules from the code. If building data is loaded by a rule engine, it still needs to be matched to the formal structure that is used to represent building regulations. This is a key aspect that is typically also considered in the approaches reported in the literature (e.g. Pauwels et al., 2011; Beach et al., 2015). In terms of the separation of rules from the code, rule language-based implementations typically keep their rules separate from the data and the generic rule engine implementation. In other words, an implementation framework is typically used (Figure 1) with a data-, schema- and rule-agnostic engine that can load data, schema and rules as required depending on user queries.

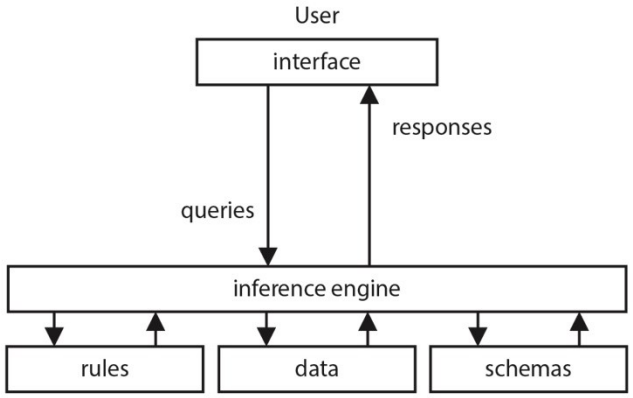

Figure 1: Outline of an implementation approach that includes a central rule engine that is kept separate from data, schema and rules (in contrast to hardcoded tools).

A number of domain-specific rule languages (first option outlined by Eastman et al., 2009) have been proposed for the construction industry. Most notable in recent proposals is the Building Environment Rule and Analysis (BERA) language (Lee et al, 2014). This language is like any domain-specific language usable only within one specific domain, in this case, the building environment, hence considerably limiting its usability and scalability, but improving its applicability and expressiveness.

On the other hand, a considerable number of general purpose rule languages (second option outlined by Eastman et al., 2009) have been proposed in the last few years, including the ones outlined in the previous section. DROOLS is indeed a rule language that could be used to capture regulatory knowledge. Yet, the formal basis of DROOLS is not that much different from the logical basis that one might find in the Java programming language. As an alternative, the use of Conceptual Graphs (CG), which have a basis in First Order Logic (FOL), has recently been proposed (Solihin, 2015). This approach offers a much more solid logical basis, but one has to keep in mind that FOL is undecidable and might not be usable if adopted incorrectly. Another example can be

found in the context of the semantic web (Berners-Lee et al., 2001), which relies at its core on a Description Logic (DL) basis (Baader & Nutt, 2003).

In the remainder of this paper, we will look into this alternative language-based approach and investigate to what extent it is possible to use it for an example regulatory document, namely the C/VM2 compliance document of the New Zealand Building Code (NZBC). This example and initial resources that are available for this compliance document are outlined and documented in Section 3, after which Section 4 and 5 indicate how this regulatory document can be encoded into language-based computable forms.

3. THE C/VM2 DOCUMENT OF THE NEW ZEALAND BUILDING CODE (NZBC)

Building construction activities in New Zealand are controlled by the New Zealand Building Code (NZBC), which is contained within the Building Regulations made under and in accordance with the Building Act 2004, the primary legislation for the domain.

NZBC is a performance-based building code specifying a set of performance criteria which the building must satisfy throughout its service life. The current edition of NZBC contains 37 technical clauses in 7 categories covering different aspects of the building design and occupancy (Table 1).

Table 1: Technical clauses of NZBC

Categories	Clauses	Remarks
Stability	B1, B2	Structure, durability
Fire safety	C1 to C6	Prevention of fire occurring, fire affecting areas beyond the fire source, movement to place of safety, access and safety for firefighting operations, structural stability
Access	D1, D2	Access routes, mechanical installation for access
Moisture	E1 to E3	Surface water, external moisture, internal moisture
Safety of users	F1 to F8	Hazardous agents on site, hazardous building materials, hazardous substances and processes, safety from falling, construction and demolition hazards, visibility in escape routes, warning systems, signs
Services and facilities	G1 to G15	Personal hygiene, laundering, ventilation, airborne and impact sound, natural light, electricity, piped services, gas as an energy source, water supplies, solid waste, etc.
Energy efficiency	H1	Energy efficiency

There are three means for a building design to comply with the NZBC, as follows:

- Acceptable Solutions, i.e. by complying fully with rules prescribed by a set of compliance documents
- Verification Method, i.e. by designing in accordance with prescribed calculations and verification methods
- Alternative Solutions, i.e. any design method (e.g. in accordance with certain standards) that can be proven to comply with the performance criteria

A case study on the compliance audit of performance-based fire safety design of buildings was conducted in a recent research project (Dimyadi et al., 2014a). In particular, the compliance document C/VM2, which is the largest set of Verification Method for Clauses C1-C6 of NZBC for the compliant fire safety design of buildings, was selected for the case study. C/VM2 contains internationally accepted performance-based fire engineering design methods, which are mostly analytical in nature and include requirements for interfacing with external computations and simulations. The general structure of the C/VM2 document is shown in Figure 2 and consists of four parts, as follows:

- Part 1 gives a list of referenced standards, definitions of terms, and an introductory section describing the application scope of the document
- Part 2 contains various prescribed rules and design parameters presented in tabular forms, mathematical equations, or embedded in paragraph texts
- Part 3 contains prescribed design parameters and rules specifically for calculations related to the movement of people

- Part 4 contains the specification of ten different fire scenarios, rules and design parameters to use in each scenario, and methods of assessing each of them for compliance

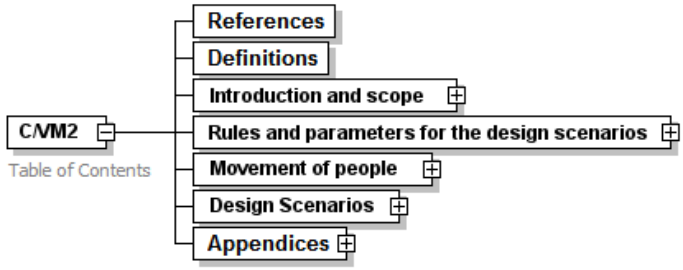

Figure 2: High-level structure of the C/VM2 paper-based document

4. MODELLING REGULATORY KNOWLEDGE USING THE RKM (REGULATORY KNOWLEDGE MODEL) APPROACH

4.1 Regulatory Document Model (RDM) and Regulatory Knowledge Model (RKM)

Research in the legal domain over the last two decades resulted in a number of useful initiatives for digitally sharing parliamentary, legislative, and judiciary documents. These include CEN's (European Committee for Standardisation) MetaLex, the United Nation's Akoma Ntoso (Architecture for Knowledge-Oriented Management of African Normative Texts using Open Standards and Ontologies) that is currently being standardised by OASIS (Organisation for the Advancement of Structured Information Standards) into LegalDocML (Vitali & Zeni, 2007); LegalRuleML (Palmirani et al, 2011; Athan et al., 2013) and LKIF (Legal Knowledge Interchange Format), which focus on the semantics and logical content of these documents.

There are generally four aspects to any document, namely presentation, structure, content, and semantics. The intent of the current standardisation process by OASIS (in the context of regulatory information) is to promote the use of:

- LegalDocML to represent the structural, literal content and presentation aspects, and
- LegalRuleML (Palmirani et al, 2011), which is based on the open standard RuleML (Boley et al, 2010) to represent the logical content and semantics of the document.

For the purposes of discussions in this paper, we refer to LegalDocML-compliant representations as RDM (Regulatory Document Models) and LegalRuleML-compliant representations as RKM (Regulatory Knowledge Models). The custom structure of both RDM and RKM is defined in XSD (XML Schema Definition). The resulting XSDs should be able to represent any regulatory document including those recommendatory in nature such as standards and any requirement specifications. An interim combined RDM+RKM model was developed to illustrate the benefits of using open standard computable forms and their roles in automating the conventional compliant design practice using examples from a selected regulatory document (Dimyadi et al. 2014a).

The combined RDM+RKM schema (in XSD) was then used to define the structure, literal and logical content of the C/VM2 document in a structured XML file that represents the metadata and content of the original regulatory document considered in this case (C/VM2). Other regulatory documents can also be represented as individual RDM whilst retaining their original structures. This approach helps maintaining user familiarity with the documents, which has the advantage of allowing these digital representations to be more seamlessly integrated into the compliant building design practice.

A high-level example of the resulting C/VM2 document structure and its content (cvm2.xml) is shown in Figure 3. Most of the RDM part of this combined model represents a sub-schema of the LegalDocML main schema. In general, the presentation aspect of RDM such as the font-style and other formatting matters can be managed relatively easily by rendering it using a standard stylesheet definition language such as XSL (eXtensible Stylesheet Language).

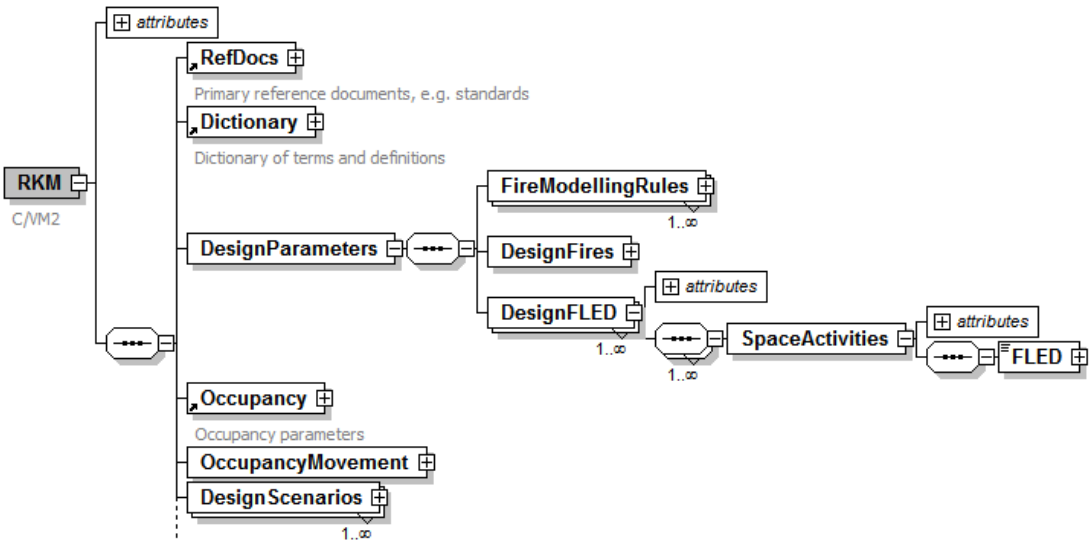

Figure 3: High-level schema of RKM for C/VM2 document for fire engineering design in New Zealand

The instance CVM2.xml file, which is based on the RKM schema shown in Figure 3, contains the following information (also see Figure 2 for an overview of the initial structure of the C/VM2 document):

- Metadata of the C/VM2 document such as version number and publication dates
- *RefDocs* is a list of referenced standards or external documents that are considered as an extension to the content of C/VM2
- *Dictionary* is a list of defined terms and concepts as used in the rules and design parameters
- *DesignParameters* relates to Part 2 of C/VM2 and contains rules and calculations input data in tabular forms, mathematical equations, or embedded in paragraph texts
- *Occupancy* gives a list of prescribed occupant load density (in persons per square m) for the calculation of potential occupant load in a space
- *OccupancyMovement* relates to Part 3 of C/VM2, which is a section that provides rules and parameters for calculations related to the movement of people
- *DesignScenarios* relates to Part 4 of C/VM2 and gives a specification of ten different fire scenarios that need to be evaluated for compliance verification.

Different regulatory documents may use different terminologies and classifications for identical objects. For example, space functions or activity types may be described differently, while referring to the same activity, so these terms need to be translated into a consistent set of codes using the same standard classification. For example, the open standard Omniclass (CSI, 2012) classification of spaces by function may be used. Hence, this classification system forms one of the important sources behind the specification of the RKM shown in Figure 3.

Rules are represented in RKM as shown in Figure 4. Each rule has an identifier (ID), a condition (the LHS of any rule), and an action (the RHS of any rule).

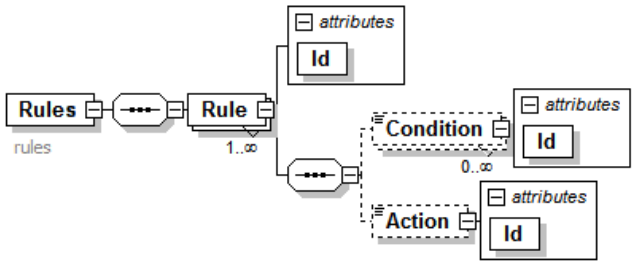

Figure 4: Rule representation schema in RKM expressed in XSD

As an example, the C/VM2 document (paragraph 3.2.3 and Table 3.3 of C/VM2) specifies the pre-evacuation time that needs to be taken into account when calculating the required evacuation time for different building uses

and locations of the fire. Pre-evacuation time is the time delay to start an evacuation and is a generally accepted concomitant event to any emergency evacuation. In the case of a public space in a retail shop occupancy (e.g. OmniClass Number 13-55 19 00 (OCCS, 2012)) where occupants are considered to be unfamiliar with the building, the specified pre-evacuation time for use in calculations is 60 seconds where the fire is in the same room as the occupants. This rule can be represented in LegalRuleML as shown in Listing 2.

Listing 2: An example rule in RKM, expressed in LegalRuleML

```
<lrml:ConstitutiveStatement key="ID_3.2.3_R1.K01">
   <ruleml:Rule key="ID_3.2.3_R1">
      <ruleml:if>
         <ruleml:And>
            <ruleml:Atom key="ID_3.2.3_R1.C1">
               <ruleml:Rel>spaceActivityCode</ruleml:Rel>
               <ruleml:Con>13-55 19 00</ruleml:Con>
            </ruleml:Atom>
            <ruleml:Atom key="ID_3.2.3_R1.C2">
               <ruleml:Rel>alarmType</ruleml:Rel>
               <ruleml:Con>Standard</ruleml:Con>
            </ruleml:Atom>
            <ruleml:Atom key="ID_3.2.3_R1.C3">
               <ruleml:Rel>location</ruleml:Rel>
               <ruleml:Con>Enclosure of origin</ruleml:Con>
            </ruleml:Atom>
         </ruleml:And>
      </ruleml:if>
      <ruleml:then>
         <ruleml:Atom key="ID_3.2.3_R1.A1">
            <ruleml:Rel>preTravelActivityTime</ruleml:Rel>
            <ruleml:Con>60</ruleml:Con>
         </ruleml:Atom>
      </ruleml:then>
   </ruleml:Rule>
</lrml:ConstitutiveStatement>
```

4.2 Compliant Design Procedures (CDP)

Human input is an important feature in performance-based design where the compliance audit procedure or method and design assumptions need to be formally documented. Building designers need to specify exactly how their designs can be verified for compliance by peer reviewers or the regulatory authority. A practical approach has been developed in the current research to allow designers to describe their own compliant design procedures (CDP) and capture any tacit knowledge they have using the open standard BPMN (Business Process Model and Notation) executable workflow model (Object Management Group, 2011), which supports XML data exchange natively (Figure 5).

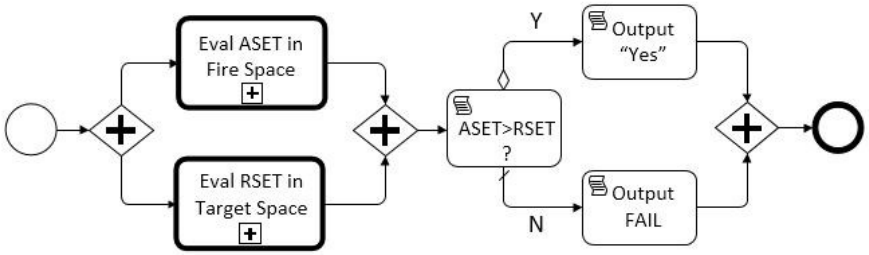

Figure 5: An example executable CDP workflow described in BPMN 2.0 (Dimyadi et al., 2014b)

This approach gives designers the freedom to explore design options that are compliant with selected regulations, while not taking away their responsibility to specify the intended compliance paths followed in their design procedures. In the context of automated compliance audit, this approach addresses issues related to the need to map objects automatically between the building model and the RDM and RKM. Each CDP workflow represents a pre-determined set of compliant design procedures specifying which objects are to be checked against which sets of requirement. Once defined, it can be executed iteratively for different design options and across multiple projects, hence automating the manual compliance audit procedures (Dimyadi et al., 2014b).

This is considered more practical than aiming for a system that attempts to derive the appropriate compliance path out of an indefinite number of options directly from regulatory texts based on a given design model. Furthermore, a CDP workflow can be used to gather the information required to generate the input data for engineering analysis or simulations. Inevitably, some information will be missing from the building model that needs to be supplemented by human input. A CDP workflow allows additional human input to be specified as necessary.

BPMN-compliant CDP workflows allow embedding of queries and instructions written in a particular computer scripting language for retrieving information from building and regulatory knowledge models. JavaScript, Groovy, Ruby, Python are common scripting languages used for business processes. These scripts can assign values to variables, and evaluate mathematical expressions or logical statements. However, the use of a standard computer scripting language requires specific knowledge and skill sets that cannot generally be expected from a building designer, hence often the need for a high-level domain-specific language that is easy to learn and use (see Section 4.3.2).

In practice, one possible scenario would be for a professional body representing the domain experts, such as the professional association of engineers, to develop a library of best practice CDP workflows. This would ensure standard workflows being used and minimise the effort required by individual designers to create their own CDPs from scratch, although designers can still modify any officially published CDP workflow to suit their own design practice or to accommodate specific design options.

A BPMN-compliant CDP workflow process engine and compliance audit tool (ARCABIM) was developed as part of the research. This has been used to successfully process a number of CDP workflows representing common design procedures to check a sample building model for compliance with a fire design scenario prescribed in the C/VM2 document.

4.3 Querying Building Information and Regulatory Knowledge

As described in Section 4.2, an executable CDP workflow can embed computer scripts to query both the building model and RKM for compliance audit processes. Representing and accessing building information is not the focus of this paper, but has been addressed in a related work (Dimyadi et al, 2016). Hence, we will only outline this briefly in Section 4.3.1 and focus mostly on accessing regulatory knowledge in Section 4.3.2.

4.3.1 Building Information and Building Compliance Model (BCM)

A building model is necessarily large and complex as it intends to capture every major object in the building throughout its life-cycle. Nowadays, building information is typically exchanged using the Industry Foundation Classes (IFC) (buildingSMART, 2016a). Accessing information directly from such a complex model would be impractical, hence the use of Model View Definitions (MVD) was proposed within buildingSMART International (buildingSMART, 2016b; Hietanen, 2008) in conjunction with the complete IFC schema. MVDs are subsets of the building model for specific applications. For example, a general MVD for compliance audit can be referred to as the Building Compliance Model (BCM). The MVD for fire safety compliant design of buildings can be referred to as the Fire Compliance Model (FCM). Similarly, the MVD for compliant electrical services design of a building can be referred to as Electrical Services Compliance Model (ESCM) and so on. It is envisaged that there would eventually be a number of MVDs associated with different design disciplines for use in practice. Each discipline-specific MVD would contain the building design information specific to that discipline in a similar way that each discipline would produce a separate set of design documentations in the conventional practice.

In our current work, the exchange requirements for FCM are actually defined in XSD. The instance FCM file of a given building (in XML) can be generated from the original IFC file using tools such as the open-source BIMserver (Beetz & van Berlo, 2010) in conjunction with a purpose-built serialiser. This FCM instance XML document would contain information specific to the requirements for the fire safety design of that building. This is necessary as the information related to the fire safety design can only be provided by the fire safety designers and as part of the design process. Such discipline-specific information would not be available, for example, from the initial architectural model.

4.3.2 Regulatory Knowledge Query Language (RKQL)

As described earlier, one way to access information in the RKM using the CDP approach is to embed computer scripts representing instructions in the CDP workflow. Designers certainly need to be familiar with the content of the RKM in order to be able to specify correct queries. This is no different from using regulatory documents to look up requirements and parameters in the course of a traditional design process or traditional compliance audit. A high-level user interface (Figure 6) may be incorporated into a compliance audit system to provide a list of objects and attributes of the models available for query or allow designers to navigate easily through the models to find the correct objects to query. Likewise, designers need to be familiar with the content of the BIM model view, so that they are able to obtain the correct building information to process.

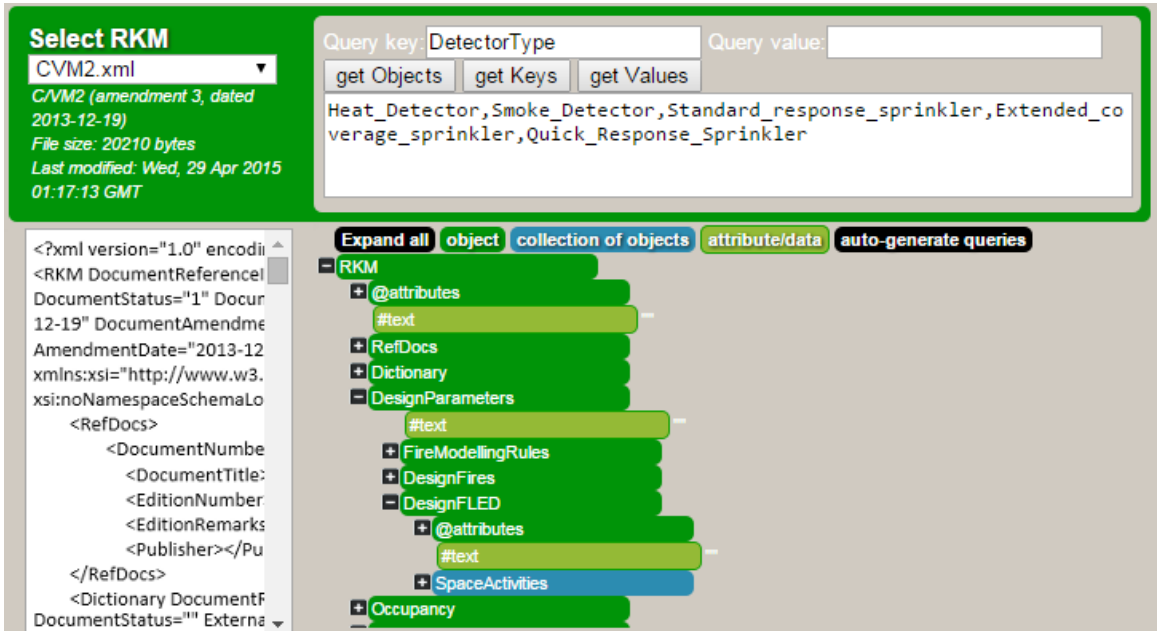

Figure 6: Example high-level user interface to help identify objects and attributes in RDM and RKM to query

It would be unreasonable to expect an end user such as a building designer to be conversant with standard computer scripting languages. For this reason, a much simpler and easy to learn domain-specific query language, referred to as Regulatory Knowledge Query Language (RKQL) was developed as part of the research. RKQL is modelled loosely on the basic querying syntax of SQL (Structured Query Language) that is commonly used with relational databases. Currently, RKQL is the default query language supported by ARCABIM. However, a separate research project is being undertaken to extend the CDP workflow capabilities by integrating an alternative domain-specific query language from another system to complement RKQL. Supporting a relatively high-level domain-specific language also has the advantage of facilitating the development of interface systems for the end user.

RKQL has been developed to hide the low-level technical functionality from the end user and provides a simple specification to aid interface system developers or building designers to write or maintain high-level scripts that can easily be embedded into the script task of the CDP workflow. RKQL mainly uses the keyword **GET** with **FROM** and **WHERE** clauses to retrieve information from the RDM, RKM, or the building model (e.g. FCM).

To describe the syntax and grammar of the language, the Extended BNF (Backus-Naur Form) notation (Figure 7) has been used. These can also be expressed as a set of syntax diagrams as shown in Figure 8. Apart from GET, the current implementation also allows EVAL and SET statements. A query to get a specific value from an object in RKM can simply be written as: **GET** *object* **FROM RKM WHERE** *condition*. By default, the type of object is assumed to be **DATA**. Optionally, RKQL allows one to specify other types of object to get, e.g. **EQUATION** or **RULE**, and to then evaluate. In a compliance audit application for a particular design discipline, the default set of RDM/RKM is usually pre-selected so that the path to its physical location is known, otherwise, their full location path may be specified (bottom right in Figure 8). To evaluate a specific rule in RKM, one

simply writes **EVAL RULE** *ruleId,* where *ruleId* is a unique ID of the rule. To set an integer or real value or a mathematical expression to a variable, one simply writes **SET** *VariableName = Integer, or RealValue, or Expression.*

```
RKQLStatement ::= getStatement | evalStatement | setStatement
getStatement ::= getClause fromClause whereClause?
getClause ::= 'GET' objectType? nameExp
fromClause ::= 'FROM' sourceExp
sourceExp ::= 'FCM' | (fullpathExp 'IN')? 'RKM'
objectType ::= 'DATA' | 'EQUATION' | 'RULE' | 'ENTITY'
whereClause ::= 'WHERE' conditionalExp
conditionalExp ::= nameExp comparisonOperator (valueExp | variable | booleanLiteral)
comparisonOperator ::= ( '=' | '!=' | '<' | '>' | '<=' | '>=')?
nameExp ::= variable | letter+
fullpathExp ::= ('/' nameExp)
valueExp ::= integer | real
setStatement ::= ('SET') nameExp '=' valueExp | expression
evalStatement ::= 'EVAL' 'RULE'? nameExp
expression ::= ('+' | '-') term+
term ::= factor ( ('*' | '/') factor )+
factor ::= integer | real | variable | function | limits | '(' expression ')'
variable ::= identifier
function ::= identifier '(' expression (',' expression) ')'
limits ::='{' expression '}'
identifier ::= letter (digit | letter | '_')+
booleanLiteral ::= 'true' | 'false'
letter ::= 'a-z' | 'A-Z'
digit ::= '0-9'
integer ::= digit+
real ::= integer real_decimals
real_decimals ::= '.' integer
```

Figure 7: Extended Backus-Naur Form (BNF) notation of RKQL domain specific query language

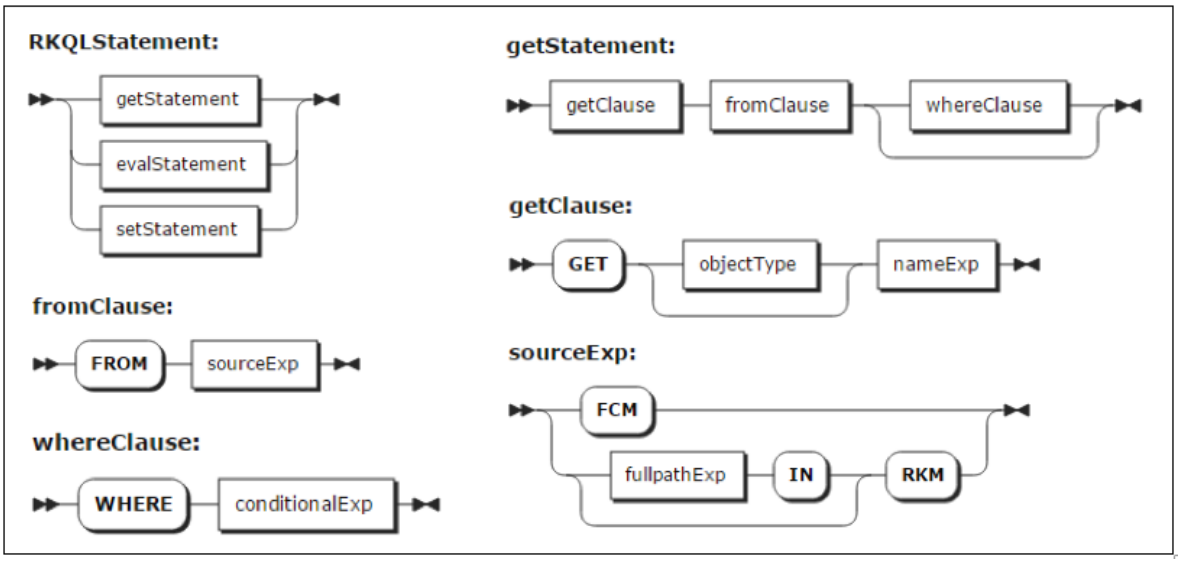

Figure 8: Selected syntax diagrams of the RKQL domain specific query language

The systematic process of an automated compliance audit of a building may be on a floor level by floor level basis starting at the top level down. Every space object on each level is then processed in turn and subject to the calculations specified in the CDP workflow. For example, given a space activity type such as "Offices" or "13-55 11 00" (in the Omniclass classification) being passed on by the variable *varSpaceActivity*, the corresponding C/VM2 prescribed FLED (Fire Load Energy Density) value for that space is 800 MJ/m^2. An example RKQL

script (embedded in a script task) to retrieve the FLED value from the RKM given a set of conditions is shown in Listing 3. Listing 4 shows an excerpt of the instance RKM document where the information to be retrieved by the query in Listing 3 is utilised.

Listing 3: Example RKQL script embedded in a script task of a BPMN-compliant CDP workflow

```
<scriptTask isForCompensation="false" id="PO_p2034" name="FLED" arcabim:unit="MJ/m^2">
   <incoming>PO_p2033</incoming>
   <outgoing>PO_p2073</outgoing>
   <script>GET FLED FROM RKM WHERE SpaceActivities.SpaceActivityCode=varSpaceActivity</script>
</scriptTask>
```

Listing 4: Excerpt of the instance RKM for C/VM2

```
<DesignFLED DocumentReferenceId="2.3.3" TableReferenceId="2.2">
    <SpaceActivities SpaceActivityCode="13-31 13 00" SpaceActivityDescription="Classrooms"
       <FLED Unit="MJ/(m^2)" Multiplier="1" MaxFLED="" MinFLED="">400</FLED>
    </SpaceActivities>
    <SpaceActivities SpaceActivityCode="13-55 11 00" SpaceActivityDescription="Offices"
       <FLED Unit="MJ/(m^2)" Multiplier="1" MaxFLED="" MinFLED="">800</FLED>
    </SpaceActivities>
    <SpaceActivities SpaceActivityCode="13-59 00 00" SpaceActivityDescription="Factory"
       <FLED Unit="MJ/(m^2)" Multiplier="1" MaxFLED="" MinFLED="">1200</FLED>
    </SpaceActivities>
```

5. MODELLING REGULATORY KNOWLEDGE USING SEMANTIC WEB TECHNOLOGIES

The previous section outlined how the C/VM2 regulatory document can be modelled using a LegalDocML-compliant RDM and LegalRuleML-compliant RKM, which is entirely XML-based; and how the resulting RDM and RKM can be accessed by CDP workflows in conjunction with query and scripting languages. This section considers an alternative approach to model and access regulatory knowledge using logic-based semantic web languages. The main difference between the two approaches is the presence of a logical framework, i.e. Description Logic (DL), which has implications for expressivity, computational efficiency and overall implementation approach. While outlining this semantic web approach, we pinpoint how this approach relates to the RDM and RKM approach documented in Section 4.

5.1 Context and Methodology

5.1.1 What languages are used for data, schemas and rules?

Semantic web technologies have their appeal in allowing the structured representation of information with ontologies and in enabling the combination or linking of disparate information sources accessible on the world wide web (Berners-Lee, Hendler, & Lassila, 2001). Furthermore, they have a basis in DL (Baader & Nutt, 2003) and include several rule languages, triple stores and reasoning engines able to respond to queries. Hence, they could also be used in the context of compliance audits for buildings.

The de facto open standard data model used by all semantic web technologies is the Resource Description Framework (RDF), a simple language originally developed to describe data or resources as targeted labelled graphs. In addition, the Web Ontology Language Language (OWL) (Horrocks, 2008) allows one to represent OWL ontologies, which are typically used to give RDF graphs a formal structure (inheritance, cardinality restrictions, range and domain restrictions). Information in the semantic web consists of triples, which are RDF expressions constructed of subjects, predicates, and objects (Figure 9). By semantically linking all kinds of objects and subjects (resources) using predicates, large clouds of Linked Data can emerge. This information is typically stored in RDF triple stores, which are a specific kind of graph database.

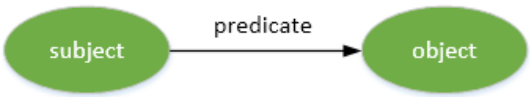

Figure 9: A triple form of an RDF expression

There are a number of open standard computing environments for managing RDF graphs. A commonly used open source Java framework is the Apache Jena, which has an API (Application Programming Interface) for reading, creating and handling RDF graphs and which supports serialising triples in a number of syntaxes, including RDF/XML and Turtle. The primary query language for RDF graphs is SPARQL (SPARQL Protocol and RDF Query Language), which is a declarative language for subgraph retrieval similar to SQL for querying relational databases (Prud'hommeaux & Seaborne, 2008).

As semantic web technologies also allow the representation of rules and the combination of such rules with available information sources, they might also be a useful set of technologies that can be used for compliance checking of building designs (Pauwels et al., 2011). The language typically used for this purpose is the Semantic Web Rule Language (SWRL) (Horrocks et al., 2004), although other semantic rule languages have been proposed and used as well, including the Rule Interchange Format (RIF) (Kifer & Boley, 2013) and N3Logic (Berners-Lee et al., 2007). Rules are typically represented and managed as separate rule sets (called the 'RBox'), in addition to the RDF data (the 'ABox') and the OWL ontologies (the 'TBox'). All three elements can be presented to an inference engine, which is able to integrate them and respond to queries using the combination of these three elements (ABox, TBox, RBox – also compare to Figure 1).

5.1.2 The logical basis

The logical basis of semantic web technologies is one of the main differences between the semantic web implementation approach and the LegalDocML- and LegalRuleML-compliant approach that is proposed in Section 4. Indeed, whereas the RDM/RKM approach relies entirely on an XML-based structure, a semantic web approach requires a domain model (the RKM) to be represented in a separate OWL ontology and associated semantic rule set.

The transition between these two worlds is not impossible. In fact, a transition has already been proposed from LegalRuleML to a Modal Defeasible Logic (MDL) representation (Lam, Hashmi, & Scofield, 2016). A similar transition could be proposed from LegalRuleML to a representation usable within a semantic web and/or linked data context. Moreover, the LegalRuleML Technical Committee (TC) agreed on a charter that aims at making LegalRuleML interoperable *"with the main languages for rule modelling, mainly Common Logic, RIF, and SWRL"* (OASIS, 2015). Also, SWRL is originally proposed by Horrocks et al. (2004) as *"a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language* [RuleML]*"*. The SWRL language has an RDF syntax and an XML syntax based on RuleML, the core of LegalRuleML.

The transition between LegalRuleML (RKM) and a semantic rule language like SWRL is not the focus of this research. Instead, we will describe the actual implementation details for a regulation compliance audit system when encoding a regulatory document like C/VM2 using semantic web languages.

5.2 The C/VM2 as an Ontology and Rule Set

5.2.1 The Regulation Ontology

An essential component of a semantic web based system is the domain ontology, which is a structured and formal representation of a particular scoped set of knowledge (the domain). The scope of the domain knowledge represented in an ontology depends on the intended application and the type of problems to be addressed. The ontology defines the object types and relations that are available for the representation of objects.

In this particular case, the domain consists of regulatory information, of which the content of C/VM2 is an example instance. Similar to how this regulatory domain was structured into a set of XSD files (Section 4.1), it was also structured into an OWL ontology based on the same set of XSD files. The knowledge inherent in the RKM thus becomes available as an OWL ontology, constituting the 'schema' node in Figure 1. A partial diagram of the resulting ontology structure is given in Figure 10.

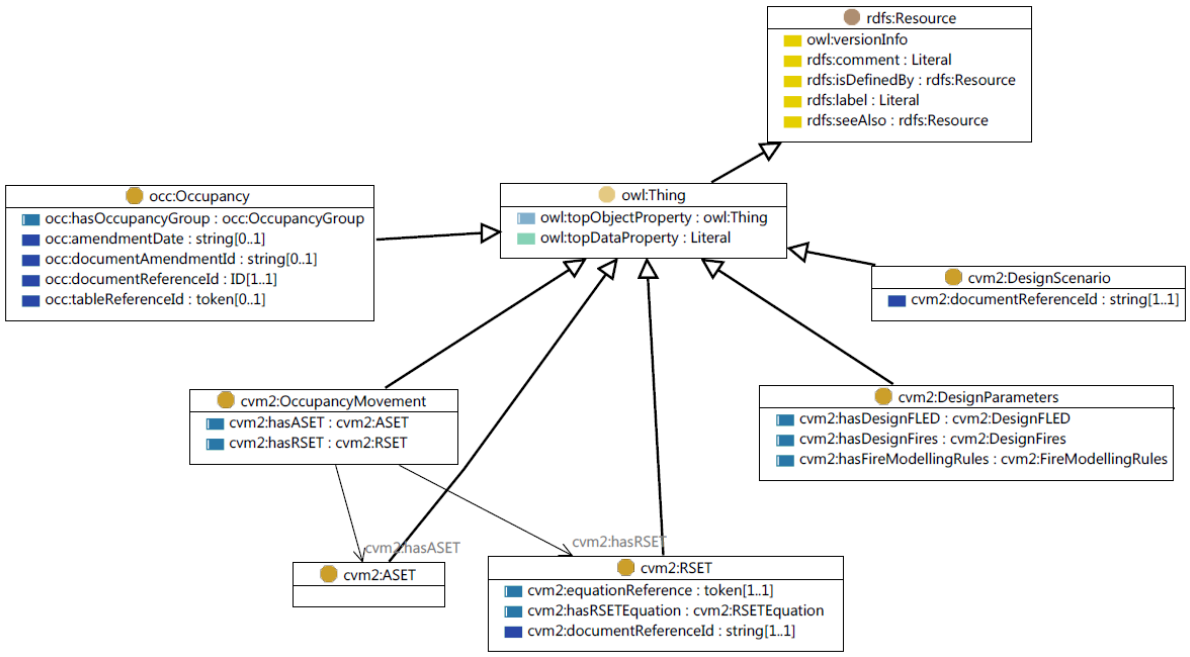

Figure 10: Partial diagram of the regulation ontology

The C/VM2 document is available in natural language text (Section 3). As the result of modelling it using RDM and RKM (Section 4.1), it is also available in XML (CVM2.xml). Similar to how the XML data instantiates the RKM XSD files, an RDF instantiation can be made of the regulation ontology in order to encode the specific C/VM2 regulation content. As an example, we can consider the excerpt of XML given in Listing 4. This data would be represented in RDF as displayed in Listing 5.

Listing 5: The representation of the XML data in Listing 4, in TTL syntax, limited to the instances `inst:designFLED_1`, `inst:SpaceActivity_1`, *and* `inst:FLED_1`.

```
inst:designFLED_1
        a RegOnt:DesignFLED ;
        RegOnt:documentReferenceId "2.3.3" ;
        RegOnt:tableReferenceId "2.2" ;
        RegOnt:hasSpaceActivity inst:SpaceActivity_1 ;
        RegOnt:hasSpaceActivity inst:SpaceActivity_2 ;
        RegOnt:hasSpaceActivity inst:SpaceActivity_3 .

inst:SpaceActivity_1
        a RegOnt:SpaceActivity ;
        RegOnt:hasSpaceActivityCode "13-31 13 00" ;
        RegOnt:hasSpaceActivityDescription "ClassRooms" ;
        RegOnt:hasFLED inst:FLED_1 .

inst:FLED_1
        a RegOnt:FLED ;
        RegOnt:hasFLEDUnit "MJ/(m^2)" ;
        RegOnt:hasFLEDMultiplier 1 ;
        RegOnt:maximumFLED "" ;
        RegOnt:minimumFLED "" ;
        RegOnt:hasValue 400 .
```

5.2.2 The Regulation Text as a Semantic Rule Set

As explained in Section 4.1, C/VM2 rules are an inherent part of the XML document (see Figure 3, Figure 4 and Listing 2). This is somewhat different in the case of a semantic web approach. In addition to the regulation ontology and C/VM2 data, a distinct rule set needs to be created for the rules using a semantic web rule language such as N3Logic, SWRL, or any other rule language. Listing 6 gives an indication of how a rule may be represented as an addition to the generated OWL ontologies for the rule that was shown earlier in Listing 2.

Listing 6: An example rule as available in a semantic web context, expressed using Manchester OWL Syntax

```
RegOnt:spaceActivityCode(?a, "13-31 13 00")
∧ RegOnt:alarmType(?a, "Standard")
∧ RegOnt:location(?a, "Enclosure of origin")
=> RegOnt:preTravelActivityTime(?a, 60)
```

Obviously, the classes and properties used in this rule set need to be compliant with the regulation ontology that is used to express the data. More precisely, the predicates `spaceActivityCode`, `alarmType`, and `location` need to be available in the regulation ontology (hence the usage of the `RegOnt:` prefix in Listing 6). If encoded as such, an inference engine is capable of parsing ontology, schema, and data (see Figure 1) and responding to user queries that are generated from a user interface.

5.3 Compliant Design Procedures (CDP) when using a Semantic Inference Engine

There are a couple of features to notice that can impact on the way in which a rule-checking process or compliance audit process is implemented using this approach. Firstly, in order for any rule to work, the LHS of that rule (the part before the => symbol in Listing 6) needs to be available and recognisable as a graph. If a graph does not contain the predicates used or the structure represented by that LHS-part, the LHS-part is never valid and the rule never fires. If the regulatory ontology is well designed, this should not be an issue. The rule can be used entirely apart from the targeted application and can be used, just like the OWL ontology and the RDF graphs compliant with that ontology, by any number of other applications (hence the advantages of flexibility and portability mentioned in Section 2.3).

A second element to notice here, is that the RHS output of the rule, in the case of Listing 6 the property `RegOnt:preTravelActivityTime(?a, 60)`, is added to the original RDF graph as soon as an inference engine executes the rule. This additional property can in that same inference run also be used by another rule of the rule set that contains it in the LHS, hence cascading through the rules in the rule set. Cascading through the rules of a rule set can occur in a forward-chaining reasoning process, or in a backward-chaining process. In the former case, the inference engine generates all information that can be logically entailed from what is given; in the latter case, the inference engine generates only the information that can be logically entailed and that responds to a particular query.

These two features have only a limited effect on the compliance audit workflow as it was proposed for the XML-based RKM in Section 4.2. In the RKM approach, it was proposed to create a set of CDP workflows (by a user or by a building authority), which defines how the available rules and data need to be combined using a BPMN approach in order to audit a design for compliance against a particular set of RKMs. The CDP workflows thus contain diverse RKQL queries of the data and rules (see example in Listing 3).

Similarly, an implementation using semantic web technologies could rely heavily on the usage of queries formulated in the standard SPARQL query language. These single queries would access the data, rules, and ontologies through a reasoning engine, as displayed in Figure 1. In fact, even the CDP workflows expressed in BPMN could also be used in this case, thus capturing in what order the SPARQL queries are meant to be fired and how data is meant to flow from one query to another. As an example, we can consider the SPARQL query in Listing 7, which corresponds to the query shown earlier in Listing 3 and which queries for the FLED value of room with a particular function (in this case the function with code `"13-31 13 00"`).

Listing 7: Example RKQL script embedded in a script task of a BPMN-compliant CDP workflow

```
SELECT ?c
WHERE {
      ?a RegOnt:hasSpaceActivityCode "13-31 13 00" .
      ?a RegOnt:hasFLED ?b .
      ?b RegOnt:hasValue ?c
}
```

5.4 Querying for compliance

Similar to what was the case for the RKM (Section 4.2), a rule set can include references to both building data and the regulation data for compliance audit processes. Representing and accessing building information is not the focus of this paper. Hence we will only briefly outline this part in Section 5.4.1 and focus mostly on the access of regulatory knowledge in Section 5.4.2.

5.4.1 Building Information

Building data that is to be used in a semantic reasoning process ideally follows a specific ontology, similar to the way in which regulation data (the C/VM2 RDF graph) should follow the regulation ontology presented in Section 5.2.1. A building ontology is readily available in the form of the ifcOWL ontology (Beetz, van Leeuwen, & de Vries, 2008; buildingSMART, 2015; Pauwels & Terkaj, 2016), which is a direct translation of the open standard IFC schema in EXPRESS as it was also presented in Section 4.3.1.

5.4.2 Regulatory Information

As described in Section 4.2, CDP workflows can embed script statements to query the RKM for rules and data. One can use the purpose-built RKQL for that purpose (Section 4.3). Similarly, SPARQL statements like the one displayed in Listing 7 can query the combination of rules, data, and ontologies. Whereas the evaluation of a specific rule in RKM occurs by calling **EVAL RULE** *ruleId*, the evaluation of a specific rule in a semantic web context occurs by querying for the RHS part of the rule. The queries can be fired directly from the user interface, implying that a software developer only needs to develop a lightweight user interface that gives an interface to the result(s) of these queries.

6. SUMMARY

In this paper, we have presented several methods commonly used to represent and access digital regulatory knowledge for compliance audit purposes. The traditional methods based on proprietary or hard-coded rule-based representations were successful in their implementations but they have the disadvantages of being costly to maintain and inflexible to changes. Many of these systems did not survive the test of time although today a few commercial tools are still adopting such 'black-box' strategies.

There is a need for an open standard regulatory knowledge representation that allows efficient access to regulatory information. We reviewed two available knowledge representations in this article: one based on CDP with LegalRuleML and the other based on semantic web technologies. Both representations and the ways in which they are used show clear similarities. In both cases, data, schemas, and rules are largely represented separately from the actual code of the code checker (language-based declarative approach); and in both cases, considerable care needs to be taken regarding the way in which the available rules and data are combined into an appropriate, consistent and complete version of the original regulation text. Regarding this latter aspect, the usage of the presented CDPs can be a very useful technique in capturing the available regulation-checking workflows.

There are also a number of important differences between both approaches. Encoding regulations using semantic web technologies results in a rule set with a particular logical basis, hence allowing the usage of data-, schema-, and rule-agnostic inference engines. Note that, while the logic-based approach may provide a way to automate some of the more established requirements and conditions, there are still a good number of aspects of regulatory compliant design that still rely on tacit knowledge and intuition, which is best handled by a human. Hence, a user interface will definitely be required in order to capture additional end-user input. Furthermore, qualitative performance-based criteria require engineering analyses, which are less amenable for a representation in a logic-based rule language. Checking such criteria thus requires the combination of the rule set and reasoning engine with complementary human input and external simulation and analysis tools.

The approach based on CDP with LegalRuleML has a formally less strict approach, as all data is encoded in XML-like structures. The research in this direction focuses more on allowing a human designer to specify exactly how compliance can be achieved by recording the procedures in a CDP workflow that can then be executed in a compliance audit system (such as ARCABIM) for multiple design options and across different projects with consistent results. For usability, a domain specific language, RKQL, has been developed to allow a

building designer or engineer to specify queries and scripts with ease and intuitively. All low-level technical specifications are hidden from the user and handled by the compliance audit process engine.

In terms of application, the compliance audit framework (ARCABIM) has been used successfully in the research to test the capabilities of executable CDP workflow approach in automating conventional compliant design processes. However, further work is required to validate the approach for scalability and effectiveness for processing more complex building models with a larger set of CDP workflows and multiple RKMs. One project is currently underway to extend the ARCABIM framework to allow CDP workflows to use another query language to complement RKQL or as an alternative to it. Another project is investigating the potential for ARCABIM to generate input data for a number of simulation tools that can provide some of the data needed to assist with the compliance audit process.

Another potential future work is to extend ARCABIM and integrate the CDP approach with the reasoning capabilities of the semantic web technology to access the regulation ontology that is modelled using open standards such as LegalRuleML and LegalDocML.

7. ACKNOWLEDGEMENTS

The first author gratefully acknowledges the funding assistance from the New Zealand Building Research Levy, which is administered by Building Research Association of New Zealand (BRANZ). The second author acknowledges the funding and support by the Special Research Fund (BOF) of Ghent University.

8. REFERENCES

Athan, T., Boley, H., Governatori, G., Palmirani, M., Pasckhe, A., & Wyner, A. (2013). OASIS LegalRuleML. In *The International Conference on Artificial Intelligence and Law*. Rome, Italy.

Baader, F., & Nutt, W. (2003). Basic Description Logics. In *The Description Logic Handbook: Theory, Implementation, and Applications*. pp. 47–100. Cambridge University Press, Cambridge, USA. doi:10.1007/s00287-011-0534-y

Beach, T. H., Kasim, T., Li, H., Nisbet, N., & Rezgui, Y. (2013). Towards Automated Compliance Checking in the Construction Industry. *Database and Expert Systems Applications*, *8055*, pp. 366–380.

Beetz, J., & van Berlo, L. (2010). bimserver.org - An Open Source IFC Model Server. In *Proceedings of CIB W78 International Conference*. pp. 1–8. Cairo, Egypt.

Beetz, J., van Leeuwen, J., & de Vries, B. (2008). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *23*(01), pp. 89. doi:10.1017/S0890060409000122

Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., & Hendler, J. (2007). N3Logic: A Logical Framework For the World Wide Web. *Theory and Practice of Logic Programming*, *8*(03), pp. 249–269. doi:10.1017/S1471068407003213

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

buildingSMART. (2015). Linked Data Working Group. Retrieved May 20, 2015, from http://www.buildingsmart.org/standards/standards-organization/groups/linked-data-working-group/

buildingSMART. (2016a). Summary of IFC Releases. Retrieved July 31, 2016, from http://www.buildingsmart-tech.org/specifications/ifc-releases/summary

buildingSMART. (2016b). Model View Definition Summary. Retrieved July 31, 2016, from http://www.buildingsmart-tech.org/specifications/ifc-view-definition

CSI. (2012). *OmniClass - A Strategy for Classifying the Built Environment Table 13 – Spaces by Function*. OmniClass.

Dimyadi, J., Clifton, C., Spearpoint, M., & Amor, R. (2014a). Computer-aided Compliance Audit to Support Performance-based Fire Engineering Design. In *Proceedings of 10th International Conference on Performance-based Codes and Fire Safety Design Methods*. Gold Coast, Queensland.

doi:10.13140/2.1.5142.7521

Dimyadi, J., Clifton, C., Spearpoint, M., & Amor, R. (2014b). Regulatory Knowledge Encoding Guidelines for Automated Compliance Audit of Building Engineering Design. In *Proceedings of the ICCCBE/CIB W78*. pp. 536–543. Orlando, Florida, USA.

Dimyadi, J., Clifton, C., Spearpoint, M., & Amor, R. (2016). Computerizing Regulatory Knowledge for Building Engineering Design. *Journal of Computing in Civil Engineering*, *C4016001*, pp. 1–13. doi:10.1061/(ASCE)CP.1943-5487.0000572

Dimyadi, J., Pauwels, P., Spearpoint, M., Clifton, C., & Amor, R. (2015). Querying a Regulatory Model for Compliant Building Design Audit. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands*, pp. 139–148.

Eastman, C., Lee, J., Jeong, Y., & Lee, J. (2009). Automatic rule-based checking of building designs. *Automation in Construction*, *18*(8), pp. 1011–1033. doi:10.1016/j.autcon.2009.07.002

Evt, S. K., Khayyal, S., & Sanvido, V. E. (1992). Representing Building Product Information using Hypermedia. *Journal of computing in Civil Engineering*, *6*(1), pp. 3–18.

Fenves, S. J., & Garrett, J. H. (1986). Knowledge based standards processing. *Artificial Intelligence in Engineering*, *1*(1), pp. 3–14. doi:10.1016/0954-1810(86)90029-4

Fenves, S. J., Gaylord, E. H., & Goel, S. K. (1969). *Decision Table Formulation of the 1969 AISC Specification*. University of Illinois Engineering Experiment Station. College of Engineering. University of Illinois at Urbana-Champaign, USA.

Hietanen, J. (2008). *IFC Model View Definition Format*. *International Alliance for Interoperability*. Retrieved from http://www.secondschool.net/one/IAI_IFC_framework.pdf

Horrocks, I. (2008). Ontologies and the semantic web. *Communications of the ACM*, *51*(12), pp. 58–67. doi:10.1145/1409360.1409377

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. Retrieved July 31, 2016, from https://www.w3.org/Submission/SWRL/

Kifer, M., & Boley, H. (2013). RIF Overview (Second Edition) - W3C Working Group Note 5. Retrieved July 31, 2016, from http://www.w3.org/TR/rif-overview

Lam, H.-P., Hashmi, M., & Scofield, B. (2016). Enabling Reasoning with LegalRuleML. In N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, & D. Roman (Eds.), *10th International Web Rule Symposium (RuleML 2016)*. pp. 241–257. New York, USA: Springer International Publishing. doi:10.1007/978-3-319-42019-6_16

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 and Robotic Systems: Theory and Applications*, *79*(3-4), pp. 507–522. doi:10.1007/s10846-014-0117-7

OASIS. (2015). LegalRuleML Technical Committee - Charter. Retrieved July 31, 2016, from http://www.oasis-open.org/committees/legalruleml/charter.php

OCCS. (2012). OmniClass Construction Classification System. Table 13: Spaces by Function.

Object Management Group. (2011). Business Process Model and Notation (BPMN), *version 2*, pp. 1–538. Retrieved from http://www.omg.org/spec/BPMN/2.0/

Palmirani, M., Governatori, G., & Rotolo, A. (2011). LegalRuleML: XML-based rules and norms. *Rule-Based Modeling and Computing on the Semantic Web*, pp. 298–312.

Pauwels, P., & Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, *63*(February), pp. 100–133. doi:10.1016/j.autcon.2015.12.003

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. doi:10.1016/j.autcon.2010.11.017

Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. *W3C Recommendation 15 Jan 2008*. Retrieved May 14, 2015, from http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

Salama, D. A., & El-Gohary, N. M. (2013). Towards Automated Compliance Checking of Construction Operation Plans Using a Deontology for the Construction Domain. *Journal of Computing in Civil Engineering*, pp. 1–51. doi:10.1061/(ASCE)CP.1943-5487.0000298

Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. *Automation in Construction*, *53*, pp. 69–82. doi:10.1016/j.autcon.2015.03.003

Vitali, F., & Zeni, F. (2007). Towards a country-independent data format: the Akoma Ntoso experience. In *Proceedings of the V legislative XML workshop*. pp. 67–86. Florence, Italy.

Zhang, J., & El-Gohary, N. M. (2013). Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *Journal of Computing in Civil Engineering*, pp. 1–39. doi:10.1061/(ASCE)CP.1943-5487.0000346

Zhong, B. T., Ding, L., 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. doi:10.1016/j.autcon.2012.06.006