Future Internet **2012**, *4*, 830-851; doi:10.3390/fi4030830

OPEN ACCESS

future internet

ISSN 1999-5903

www.mdpi.com/journal/futureinternet

Article

Semantic Web Approach to Ease Regulation Compliance Checking in Construction Industry

Khalil Riad Bouzidi [1,2], Bruno Fies [1,*], Catherine Faron-Zucker [2], Alain Zarli [1] and Nhan Le Thanh [2]

[1] French Scientific and Technical Centre for Building (CSTB), 290 route des Lucioles, BP 209, 06904 Sophia Antipolis, France; E-Mails: khalil-riad.bouzidi@cstb.fr (K.R.B.); alain.zarli@cstb.fr (A.Z.)

[2] The I3S laboratory (Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis), University of Nice Sophia Antipolis and CNRS, BP 121, 06903 Sophia Antipolis, France; E-Mails: faron@polytech.unice.fr (C.F.Z.); nhan.le-thanh@unice.fr (N.L.T.)

* Author to whom correspondence should be addressed; E-Mail: bruno.fies@cstb.fr; Tel.: +33-493-956-756; Fax: +33-493-956-733.

Received: 6 July 2012; in revised form: 23 August 2012 / Accepted: 27 August 2012 / Published: 11 September 2012

Abstract: Regulations in the Building Industry are becoming increasingly complex and involve more than one technical area, covering products, components and project implementations. They also play an important role in ensuring the quality of a building, and to minimize its environmental impact. Control or conformance checking are becoming more complex every day, not only for industrials, but also for organizations charged with assessing the conformity of new products or processes. This paper will detail the approach taken by the CSTB (Centre Scientifique et Technique du Bâtiment) in order to simplify this conformance control task. The approach and the proposed solutions are based on semantic web technologies. For this purpose, we first establish a domain-ontology, which defines the main concepts involved and the relationships, including one based on OWL (Web Ontology Language) [1]. We rely on SBVR (Semantics of Business Vocabulary and Business Rules) [2] and SPARQL (SPARQL Protocol and RDF Query Language) [3] to reformulate the regulatory requirements written in natural language, respectively, in a controlled and formal language. We then structure our control process based on expert practices. Each elementary control step is defined as a SPARQL query and assembled into complex control processes "on demand", according to the component tested and its semantic

definition. Finally, we represent in RDF (Resource Description Framework) [4] the association between the SBVR rules and SPARQL queries representing the same regulatory constraints.

Keywords: ontology; semantic web; knowledge management; building industry; e-regulations; assisted checking; rule based system

1. Introduction

Regulations in the Building Industry are becoming increasingly complex and involve more than one technical area, covering products, components and project implementations. They also play an important role in ensuring the quality of a building, and minimizing its environmental impact. As a consequence of this complexity; checking the conformity of a new product against the existing regulations is becoming more complex for industrials every day. In this general context, the research communities of the Knowledge Engineering and Semantic Web play a key role in providing models and techniques to simplify access to technical regulatory information, facilitate its appropriation, and support professionals in its implementation.

In this article, we address the different issues linked to compliancy, checking specifically the production of Technical Advice (so-called "ATec" in French). An ATec is a document containing technical information on the usability of a product, material, component of construction or even an innovative process of construction. However, this new product or new process must comply with existing regulatory documents. ATec was chosen as a study model for this project because CSTB has the mastership and a wide experience in these types of technical documents. We were able to lead interviews on the CSTB site of Sophia-Antipolis with experts directly involved in the drafting of these ATec in the field of photovoltaic panels.

2. Selected Regulatory Documents

2.1. Importance and Difficulties Linked to Regulations

The regulations are written and applied by humans and are therefore vulnerable to human error. They can at times be incomplete, contradictory, or arbitrarily complex. This has at least one direct consequence in the present scope. If we want to automate the verification process of compliancy of a new product, the textual knowledge and the corresponding constraints contained in these documents have to be translated (and transformed) into rules understandable by machines and, thus processable.

2.2. The Technical Guides

In this paper, we will illustrate our approach by not considering direct standards or regulatory documents, but Technical Guides (TG). These TGs are documents edited by the CSTB that explain regulatory documents; they do not replace the regulations. The TGs are a complement to the regulatory documents, offering the industrials easier reading and understanding of the technical rules of construction. Also, they collect detailed executions featuring a wide range of situations. The TGs are

also the documents of reference that help verify the validity of the technical information provided by the manufacturers. All of the structural and dimensional variables for the validation of a product are encompassed within the TGs. Our goal is to formalize the knowledge they contain as constraints database, exploitable by knowledge-based systems.

We use the TG "The tile roofs" issued by CSTB as a study model. This guide of 107 pages outlines the different types of tiles and their characteristics. It defines their status of implementation and verification of various criteria, such as slope tiles, support or climatology of their place of installation. Monitoring these instructions is of utmost importance because non-conformance with a requirement leads to a negative on the Technical Assessment Technical Paper.

According to the TG "The tile roofs", we were able to identify nine different types of tile, with different intrinsic characteristics. Each tile has a manufacturing area and a shape, an implementation that depends on the slope and support, and raises an attachment.

The regulatory constraints existing in these TGs are used to decide whether the procedures used by manufacturers meet their obligations.

2.3. The Technical Advice (ATec)

As already introduced in the first chapter, an ATec corresponds to an innovative product or process and it mainly contains a technical description (made by industrial wanting to promote its new product or process). The structure of an ATec is always the same and it consists of three parts:

- An overview and identification of the assessment, which could be considered as an administrative part.
- The technical assessment itself, formulated by a group of experts from CSTB. Basically, only two options are possible in this section. It should be answered clearly if the element or process described is acceptable or not.
- The technical document (TD) of the product or process which must be delivered in the technical assessment. Fulfilling this part is devoted to the industrial who wants to promote an innovative product or process. For several reasons, this is the most difficult part that requires efforts from the industrial and time from CSTBs' experts.

Therefore, we are particularly interested in assisting the creation of the TD.

An ATec is drafted at the request of an industrial. The industrial sends a request to the relevant department within the CSTB and in return, the CSTB sends back a template of the TD to the industrial. It is a Word document which contains chapters, text and instructions. This template is supposed to be self-sufficient but it is apparent that industrials fail to properly prepare the template in order to create a good TD. A dialogue between CSTB and the applicant is therefore necessary before reaching an acceptable version of the TD. As a direct consequence, this leads to a long casework, which takes approximately 6 to 8 months and requires an effort from both sides that must be reduced to a minimum.

3. Modeling Technical Documents

The regulations as well as the TD are written in a flat textual format that is understandable. The corner stone of our approach is to develop a common framework allowing structured and semantically

rich representations of regulation contents and product or process structures. After reviewing various studies related to technical regulations and after having interviewed experts involved in the elaboration of ATec, we defined a generic process for the verification of the TD. This process is a formalization of practices followed by CSTBs' experts. In the continuation of the work performed at CSTB [5], about ontology building and conformity checking of construction projects [6], we propose to model the process of elaborating the TD using an ontology-based approach.

We represent the regulatory constraints using the above mentioned ontology as the source for a controlled vocabulary and we defined a methodology to transform the rules expressed from natural language (which is the case for regulation) to semi-formal language (SBVR) and to a formal language (SPARQL). This part will be explained in Section 4 of the current article.

3.1. The OntoDT Ontology

In order to disambiguate the semantic attached to the terms; we defined an ontology of the considered domain. We defined an ontology, so-called "OntoDT", to represent both the structure of the TD but also the vocabulary used in the considered technical field.

We have studied the TDs issued by a group within CSTB that is responsible for validating the ATec concerning photovoltaic panels. The OntoDT ontology contains an exhaustive list of terms from the photovoltaic domain merged with the other terms extracted from a thesaurus developed by CSTB for the building industry which is called the REEF [7].

We developed a model for tiles based on information collected from a dedicated Technical Guide named "The tile roofs". Each tile is represented in concept and integrated into our ontology, OntoDT. The ontology includes all the semantics reflecting structural and dimensional criteria of a tile; the criteria are represented by properties.

As a result, our ontology has 138 classes and 48 properties formalized in the OWL Lite language; 35% of these classes are created from REEF terms. The remaining 65% are concepts more specific than those of the REEF thesaurus, which contains general concepts of the building industry. In its current state, it lacks specific terms relative to a particular field (Photovoltaic). However, it remains in constant evolution.

3.2. Modeling the Technical Document

We use our ontology, OntoDT, to model the semantics conveyed by the TD into a formal interpretable knowledge. We translated the template of the TD to a set of forms interconnected to each other. The product to be described is decomposed into a set of sub elements called "components". These components are identified in the OntoDT and thus linked by semantic relationships to other components. These relationships will guide the concatenation of the corresponding forms. In other words, in an application developed on this basis, the user will be guided in the process according to the information entered in the forms. The way the forms follow on from each other is determined by the ontology and by data entered by the user.

For example, a PV glass polymer module (a component) is part of a PV panel (the product). This glass polymer module is composed of several elements: polymer film, photovoltaic cell, *etc.* (elements).

To model this composition, each concept representing a component, an element or a product is defined by an axiom. Axioms are used in the definition of an ontological class, they are of the A ⊆ B form, where A is a primitive concept (Product) and B description composed concept. For instance, (Figure 1) provides the definition of the class PolymerGlass in the OWL language: it is a subclass of the class PVPanel and of a class defined as the intersection of the classes of the instances having components of classElectricCable, Frame, PVCell, *etc*.

Figure 1. Example of owl code used to define a concept.

```
<owl:Class rdf:about="#PolymerGlass">
 <rdfs:subClassOf rdf:resource="#PVPanel"/>
 <rdfs:subClassOf>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasComponent"/>
        <!-- ElectricCable -->
        <owl:someValuesFrom rdf:resource="&Reef;#01573"/>
      </owl:Restriction>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasComponent"/>
        <!-- Frame -->
        <owl:someValuesFrom rdf:resource="&Reef;#01593"/>
      </owl:Restriction>
          ...
    </owl:intersectionOf>
   </owl:Class>
 </rdfs:subClassOf>
</owl:Class>
```

We have described the interdependencies between the different concepts modeling the interactions or interdependencies between parts of PV modules. These have a number of intrinsic characteristics (length, weight, manufacturer, *etc.*) represented in our modeling by properties attached to the corresponding concepts.

Once the industrial fills in the form relative to a concept with all the elements relative to its definition, they are stored in an RDF annotation file. By doing so, we acquire an interoperable representation of a TD, reusable in other systems. The use of the RDF model will allow us to later to check the conformity of the TD with the standards of the photovoltaic domain.

3.3. Modeling the Process of Writing a Technical Document

The question which now arises is how to model the process itself of completing the TD, in order to produce a dynamic sequence of forms to fill in. The dynamicity of the forms depends on the way

previous forms are filled out; we want to adjust to the information provided without requiring the industrial to complete irrelevant parts of the TD.

Our approach is to initially ask for high-level information (name the type of product, *etc.*) then browse the explicit dependency rules in the ontology to seek all information required for a product. From the information provided by the industrial, the domain ontology is used to determine the next iteration. The industrial will have to, in an initial form, choose a product among all those concerned with the photovoltaic field. Using this information, based on our ontology, we determine the list of components used in its manufacture and we offer a list of corresponding forms.

The dynamics of the sequence of forms relies upon a SPARQL query pattern presented in (Figure 2) which we initiate to query the ontology to determine the next form. More precisely, by using this query template, we query a product on its composition by questioning its definition. The concepts involved in its definition are returned by the query and necessitate more information that must be provided by the industrial through entry forms generated on the fly. The chaining of forms thus depends on the query results: each form corresponds to one or more elements of the result.

Figure 2. Extract of the SPARQL query pattern.

```
SELECT ?Component WHERE{
?x rdfs:subClassOf ?y
FILTER( ?x=dt:Component-Name)
?y owl:intersectionOf ?z
?z rdf:rest*/rdf:first ?f
?f owl:onProperty ?p
?f owl:someValuesFrom ?Component}
```

For example, the query below searches the ontology on the definition of the concept "PolymerGlass". First, this query pattern checks if the component contains one or several elements (rdfs:subClassOf, owl:intersectionOf). Second, it iterates all the elements corresponding to the nomenclature of the component (rdf:rest*/rdf:first). Third, it returns all the elements as results (owl:someValuesFrom).

```
SELECT ?Component WHERE{
?x rdfs:subClassOf ?y
FILTER( ?x=dt:PolymerGlass)
?y owl:intersectionOf ?z
?z rdf:rest*/rdf:first ?f
?f owl:onProperty ?p
?f owl:someValuesFrom ?Component
```

The result is that "PolymerGlass" is a module which has as components a "Frame", a "PVCell", *etc.*

Result: [Frame, PVCell, PolymereFilm, InnerGlass…]

At each step of the process of writing a TD, we display a form to the industrial for entering information related to a concept belonging to the result of such a query. If this concept is already defined, the same query template is initiated with a new query in order to provide the industrial with new forms matching the components involved in the definition of the current concept. The same query pattern is recursively instantiated until reaching terminal concepts, *i.e.*, primitive concepts (with no definition).

As a result, the industrial browses our ontology to complete all components of its product by filling out the forms provided; only relevant questionnaires are displayed.

3.4. Generation of Structured Annotations

To validate our approach, we developed a tool to assist the production of TD. This application has been developed in J2EE. It conforms to the three-tier architecture. The first third consists of a web page on the client side (GUI) that supports various forms to fill in by the industrial. The second third on server side is a SERVLET [8] containing a business code that interacts with the GUI part and the business part. It is connected to the KGRAM [9] semantic engine to query the ontological knowledge that represents the TD and generates the RDF model of the filled TD. The last third is relative to the data. It includes the ontology of the TD and information of the product stored as RDF annotations.

Our tool provides the industrials with a rich and interactive interface based on a sequence of forms. All along the drafting process, the industrial is guided in his choices by the OntoDT ontology and the system adapts and goes through the ontology to seek progressively more specific information. These sequences of forms are orchestrated through SPARQL queries that run on the ontology.

At the end of the sequence of forms, two files are generated: a human readable file containing the information filled in by the industrial, and a semantic description of the document in RDF. This will be exploited to help in writing the technical assessments itself, by automatically checking the conformity of the information in the TD with the regulatory texts of the domain.

4. Modeling Business Rules

Within the vast field of regulatory modeling, one standard fits our problem particularly well. Focussing on the transformation of regulatory documents [10–12], this standard is the SBVR standard. SBVR stands for “Semantics of Business Vocabulary and Business Rules”. It is an Object Management Group (OMG) [13] standard whose ultimate objective is to provide a meta-model that allows establishing data exchange interfaces for tools that create, organize, analyze and use vocabularies and business rules [14–15]. The SBVR meta-model facilitates the validation, analysis, alignment, and fusion of business rules for different tools of different constructors. The development of an SBVR base is done in two steps: the development of a business vocabulary and the writing of business rules based on the terms and concepts defined in the vocabulary.

SBVR controlled vocabularies consist of hierarchies of concepts specific to a certain domain, their relationships, definitions and synonyms. SBVR rules are based upon the Predicate Logic: they capture the “what” of business rules, rather than the “how”, in other words the semantics of business rules, not the way they must be executed. SBVR is not an executable formalism; it is particularly addressed to

business experts. It uses a controlled natural language that all business experts understand. It does not have a specific rule format.

In this paper, our SBVR examples are presented following the SBVR-based Structured English [14], using several font styles:

- **Term**: the "**term**" font is used to represent object types, concepts, (e.g., **PolymerGlass**, **PVCell**, **Module**).
- ***Verbs***: the "***verb***" font is used to represent a verb, a preposition, or a combination of these two (e.g., **PolymerGlass** ***is*** **a** **module**).
- **Keyword**: the "**keyword**" font is used for linguistic symbols used to construct statements. These keywords are combined with **terms** and ***verbs*** to create the rules (e.g., **each** **module** ***has*** **at least a** **PolymereFilm**).

4.1. Transforming Standards into SBVR Rules

There are various approaches to detect whether a sentence may describe a business rule [16]. However, technical standards can be understood in different ways that is why the manual intervention of a domain expert is essential. We argue that Natural Language Processing (NLP) approaches of knowledge extraction from regulatory texts can significantly alleviate the task of domain experts but cannot replace them. In our work, we do not consider linguistic analysis of texts but focus on the representation of expert knowledge. CSTB experts helped us to identify and classify the constraints expressed in the photovoltaic standards and then the rules which represent them. The goal of this categorization is to determine the levels of interoperability of each sentence or paragraph of the standards and classify it.

Also, despite the intervention of a domain expert to identify the meaning of the constraints, some of them remain non-interpretable. These types of constraints contain information that is too ambiguous and impossible to formalize.

Once these texts are identified, a step of disambiguation is necessary. The transformation of texts into SBVR rules will provide a normative, unambiguous and reusable source.

The extraction of rules from standards or statutory text is a tedious job; it often requires structuring the information. The descriptions used in our work have been detailed enough to show how the content of standards can be converted into SBVR vocabulary and business rules. However, a clarification of the text was needed before the transformation into SBVR. The steps below are necessary in order to produce an understandable SBVR text:

4.1.1. Extraction of Conformance Rules from Table

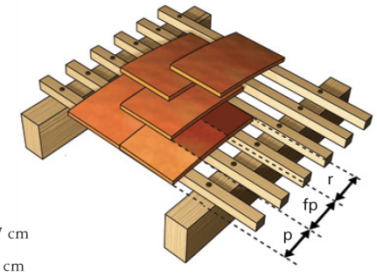
Let us start with the following Technical Guide table (Figure 3):

Figure 3. Extract of Technical Guide (TG) "The tile roofs".

(**Extrait du** *Guide Pratique Les couvertures en tuiles* © **CSTB juin 2010.** http://boutique.cstb.fr)

■ Pentes et recouvrements des tuiles plates

Pose sans écran
DTU 40.23, tableau 1



Emploi admis et recouvrement « r » ≥ 7 cm
Emploi admis et recouvrement « r » ≥ 8 cm
Emploi admis et recouvrement « r » ≥ 9 cm
Emploi non admis

r : recouvrement
fp : faux pureau
p : pureau

Pentes valables pour les rampants de longueur maximale de 8 m (en projection horizontale)

Pente de couverture	Zone I (1)			Zone II (1)			Zone III (1)		
	Situation (1)			Situation (1)			Situation (1)		
	Protégée	Normale	Exposée	Protégée	Normale	Exposée	Protégée	Normale	Exposée
70 %	8 cm			8 cm					
75 %	8 cm			8 cm					
80 %	7 cm	8 cm		7 cm			9 cm		
85 %	7 cm	8 cm		7 cm			9 cm		
90 %	7 cm	7 cm		7 cm	8 cm		8 cm		
95 %	7 cm	7 cm		7 cm	8 cm		8 cm		
100 %	7 cm	7 cm	8 cm	7 cm	7 cm		8 cm	9 cm	
105 %	7 cm	7 cm	8 cm	7 cm	7 cm		8 cm	9 cm	
110 %	7 cm	7 cm	7 cm	7 cm	7 cm	8 cm	8 cm	8 cm	
115 %	7 cm	7 cm	7 cm	7 cm	7 cm	8 cm	8 cm	8 cm	9 cm
120 %	7 cm	7 cm	7 cm	7 cm	7 cm	7 cm	8 cm	8 cm	9 cm
≥ 125 %	7 cm	7 cm	7 cm	7 cm	7 cm	7 cm	8 cm	8 cm	8 cm

1. Les zones et situations de concomitance vent/pluie sont définies dans le chapitre « Climatologie », page 108.

The presented page above, extracted from the Technical Guide "The tile roofs" is a typical example of the difficulties that can be encountered when translating information expressed in natural language to explicit and processable assertions. The example above contains a table with different information relative to the installation of tiles with their dimensional and structural properties. The first column shows different applicable slopes. In each row, we can see for a specific slope, roof covering, and area of installation. There are three different so-called "climatic zones" in France (From Zone I to Zone III, mainly based on the average temperature over a year). Each of these zones is divided into three sub categories according to the local situation of the concerned roof. It depends if the roof is exposed or not to the wind or the rain ("protected"/"normal"/"exposed").

To transform these constraints into SBVR rules we process them in two steps:

First, we transform the information in rows into textual constraint. Example, from the first row we can extract the information below:

The applicable tiles slope for a roof covering greater than 8 cm, built in zone 1, in a protected situation is equal to 70%.

The second step is to rewrite this text into SBVR rules by using the ontology of TDs, the text will read as follows:

- **If the tile *has* a slope equal to 70% then it is obligatory that the implementation is in Zone 1, in a protected situation with a roof covering greater than 8 cm.**

The concepts identified in this fragment are Slope, Zone1 and protected, which belong to the ontology of TD.

4.1.2. Reformulation

Let us consider the following regulatory text:
The dimension of the main frame must be:

- Width: (847 ± 5) mm.
- Height :(1910 ± 5) mm.

This standard extract expresses conditions that are difficult to read by non-expert readers. It needs a reformulation to be understood: "The maximum width of a main frame must be lower or equal to 853 mm and the minimum width higher or equal to 842mm. The maximum height of a main frame must be lower or equal to 1915 mm and the minimum height greater than or equal to 1905 mm".

- **If a frame *has* a minimum width higher or equal to 842mm and *has* a minimum height higher or equal to 1905mm and *has* a maximum width less than or equal to 853mm and *has* a maximum height less than or equal to 1915mm, then it is a main frame**

4.2. Transformation SBVR

Our SBVR rules are Operative Business Rules with "If-then" syntax. These kinds of rules can be transformed into systems rules, automated execution of business processes and allow checking how business activities are conducted.

However, the TG's constraints are specific to a relevant situation of a constraint to check. Our goal is to check the conformance of TDs according to constraints expressed in the Technical Guide.

So, we built a new RDF annotation (Figure 4) based on abstract language (Section 5.1).

Figure 4. The RDF annotations of if-then SBVR rules.

```
<rdf:RDF >
        <Test>
                                <if>
                                        <Antecedent>
                        </if>
                            <then>
                                        <Consequent>
                            </then>
        <Test>
        …
</rdf:RDF>
```

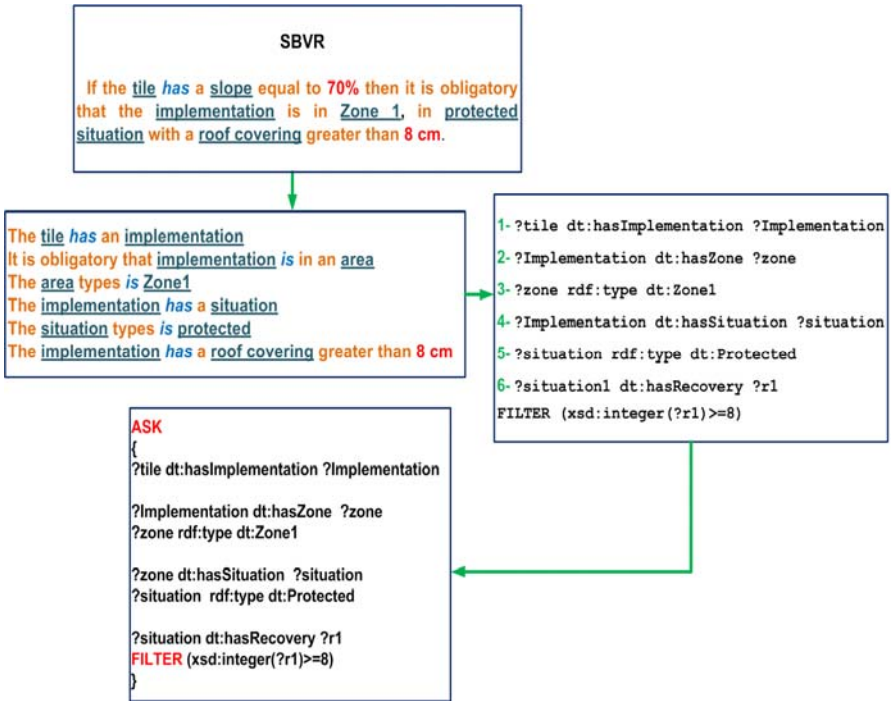
For a specific condition, it allows executing a relevant action using queries representing the Antecedent (which is in the "If" part of the query) and Consequent (which is expressed in the "Then" part of the query, or the action to be performed if the Antecedent is verified).

We represented the Antecedent and Consequent part with SPARQL "ASK" queries, this query model determines if there is a matched triple between the query and the RDF annotation of the TD. "ASK" query returns "TRUE" for positive response and "FALSE" if there is no match. Using this RDF annotation allowed us to automate a part of the process of conformance check.

4.3. Representation of Regulatory Constraints in SPARQL

We aim to model the way experts use CSTB guides and try to automate their know-how. It forces us to follow their interpretation and to establish a formal representation of regulations. SBVR describes the concepts and requirements regardless of their implementation. We chose to represent constraints into "ASK" SPARQL queries. This formalization allows the automation of the conformance of a document against these constraints (see Figure 5).

Figure 5. Transforming SBVR into SPARQL queries



To perform a transformation of SBVR rules into SPARQL queries we followed a number of defined steps and used a controlled-vocabulary, in our case we used OntoDT.

As an example the following SBVR rules:

- **If the tile *has* a slope equal to 70% then it is obligatory that the implementation is in Zone 1, in protected situation with a roof covering greater than 8 cm**

Step 1: We decompose the SBVR rule into a set of single sentences in order to decompose the different constraints expressed. So we can identify the antecedent and the consequent of our "if-then" rules as follows (Table 1):

Table 1. Decomposing SBVR rules into Antecedent and Consequent.

Rules parts	English
Antecedent	The tile *has* a slope The slope equal to 70%
Consequent	The tile *has* an implementation It is obligatory that implementation *is* in an area The area types *is* Zone1 The implementation *has* a situation The situation types *is* protected The implementation *has* a roof covering greater than 8 cm

Step 2: We transform, into triple pattern, each formulation expressed in the antecedent and consequence. We used all concepts and properties of OntoDT (Table 2).

Table 2. Transforming SBVR rules into SPARQL triple pattern.

SBVR	SPARQL Triple pattern

The tile *has* a slope
The slope equal to 70%
	minus {
	?tile dt:hasSlope ?slope
	FILTER (xsd:integer(?slope) != 70)
	}

The tile *has* an implementation	?tile dt:hasImplementation ?implementation
It is obligatory that implementation *is* in an area	?implementation dt:hasZone ?area
The area types *is* Zone1	?area rdf:type dt:Zone1
The implementation *has* a situation	?implementation dt:hasSituation ?situation
The situation types *is* protected	?situation rdf:type dt:Protégé
The implementation *has* a roof covering greater than 8 cm	?situation1 dt:hasRecovery r1 FILTER (xsd:integer(?r1)>=8)

Step 3: We build the SPARQL queries (Figures 6,7):

Figure 6. SPARQL query of the antecedent (Example 1).

```
PREFIX dt:<http://www.semanticweb.org/DossierTechniqueProtegeV.owl#>
ASK
{
?tile dt:hasImplementation ?implementation
minus {
?implementation dt:hasSlope ?slope
FILTER (xsd:integer(?slope) != 70)
}
}
```

Figure 7. SPARQL query of the consequent (Example 1).

```
PREFIX dt:<http://www.semanticweb.org/DossierTechniqueProtegeV.owl#>
ASK
{
?tile dt:hasImplementation ?implementation

?implementation  dt:hasZone ?zone
?zone rdf:type dt:Zone1

?zone dt:hasSituation ?situation
?situation rdf:type dt:Proteger

?situation dt:hasRecovery ?r1
FILTER (xsd:integer(?r1)>=8)
}
```

Step 4: Construction of the RDF annotation of the SBVR (Figure 8).

Figure 8. RDF annotation of the SBVR (Example 1)

```
<rdf:RDF >
        ...
        <Test>
                                <if>
                                        <Query rdf:about="&OntoDT;[1]#P70"/>
                        </if>
                            <then>
                                        <Query rdf:about="&OntoDT;#Check70"/>
                             </then>
        </Test>
                    ...
</rdf:RDF>
```

5. Modeling of the Verification Process of Regulatory Constraints

5.1. Process Model

We proceeded with interviewing our experts, who are responsible for managing ATec in order to capture their knowledge and then to propose a process model representing the way they work. Our model enables building an RDF description of the sequence of constraint verifications performed to control a given TD.

The RDFS schema of our model comprises four properties: body, if, then and or else, and eight classes: Pipeline, Pipe, Load, Query, Rule, RuleBase, Test and And, among which the class Pipeline models a process definition and the class Pipe models a call for a process. The execution of queries (Query) or rules (Rule, RuleBase) can be conditional (Test) and a process description can recursively call for other processes (Pipe), including itself. This recursive feature is used in the modeling of complex processes calling for one or several elementary processes.

The abstract syntax of a process is defined by the following grammar:

```
Pipeline  ::= EXP +
EXP       ::= Load(Name) | Query(Name)
            | Test(Query(Name), Exp, Exp)
            | Rule(Name) | RuleBase(Name)
            | And(Exp +) | Pipe(Name)
```

We have developed a process engine based on the KGRAM semantic engine [9] with the following principle: it analyses a process definition represented in our model and dynamically constructs and executes a sequence of SPARQL queries or rules. It thus enables supervision, coordination and sequential execution a set of queries and rules. The process management relies on a set of predefined SPARQL queries such as the one presented in (Figure 9), dedicated to the management of the body of a process, which enables listing all the components of a process and their types:

Figure 9. A SPARQL query template to interpret a process RDF representation.

```
SELECT * WHERE {
?p rdf:type kg:Pipeline
{
?p kg:body ?q
?q rdf:type ?t
minus {?q rdf:type rdf:List}
}
UNION
{?p kg:body ?a
?a rdf:rest*/rdf:first ?q
?q rdf:type ?t }}
```

For instance, let us consider the validation process which RDF representation is presented in (Figure 10). Once this RDF data is loaded, our process engine executes the above query template (Figure 9) the resource of class Pipeline is identified and the resources denoting sub-processes involved in its definition are listed. Each of these sub-processes is recursively handled in order to identify the operations to be performed. Note that we do not use the whole process model since our process representations do not involve rules: in our case, the only basic operations are queries.

Figure 10. Extract of the RDF representation of a verification process

```
<rdf:RDF >
    <Pipeline>
        <body rdf:parseType="Collection">
        <Load rdf:about="DocumentTechnique.rdf"/>
            <Test>
                <if>
                    <Query rdf:about="&OntoDT;#P70"/>
                </if>
                <then>
                    <Query
                rdf:about="&OntoDT;#Check70"/>
                </then>
            </Test>
            ...
        </body>
    </Pipeline>
</rdf:RDF>
```

Let us detail the handling of such a RDF description. The process engine interprets a resource of type Load (line 4) by loading the RDF description of a TD (located at the URI in argument) whose validity is to be checked. It interprets a resource of type Query (line 7) by loading the SPARQL query located at the URI and executing it on the loaded RDF description of a TD.

A resource of type Test (line 5) calls for the instantiation and execution of the query template presented in (Figure 11) which enables identifying the value of the "if, then" and "or else" properties and their types. The process engine then first executes the ASK query denoted by the value of the "if" property and, depending on a TRUE or a FALSE answer, it recursively interprets the RDF description of the process denoted by the value of the "then" property or by the value of the "or else" property, if any.

Figure 11. A SPARQL query template for identifying conditional sub-processes.

```
SELECT DEBUG * WHERE {
?q kg:if ?qi
OPTIONAL {?q kg:then ?qt . ?qt rdf:type ?tt}
OPTIONAL {?q kg:else ?qe . ?qe rdf:type ?te}}
```

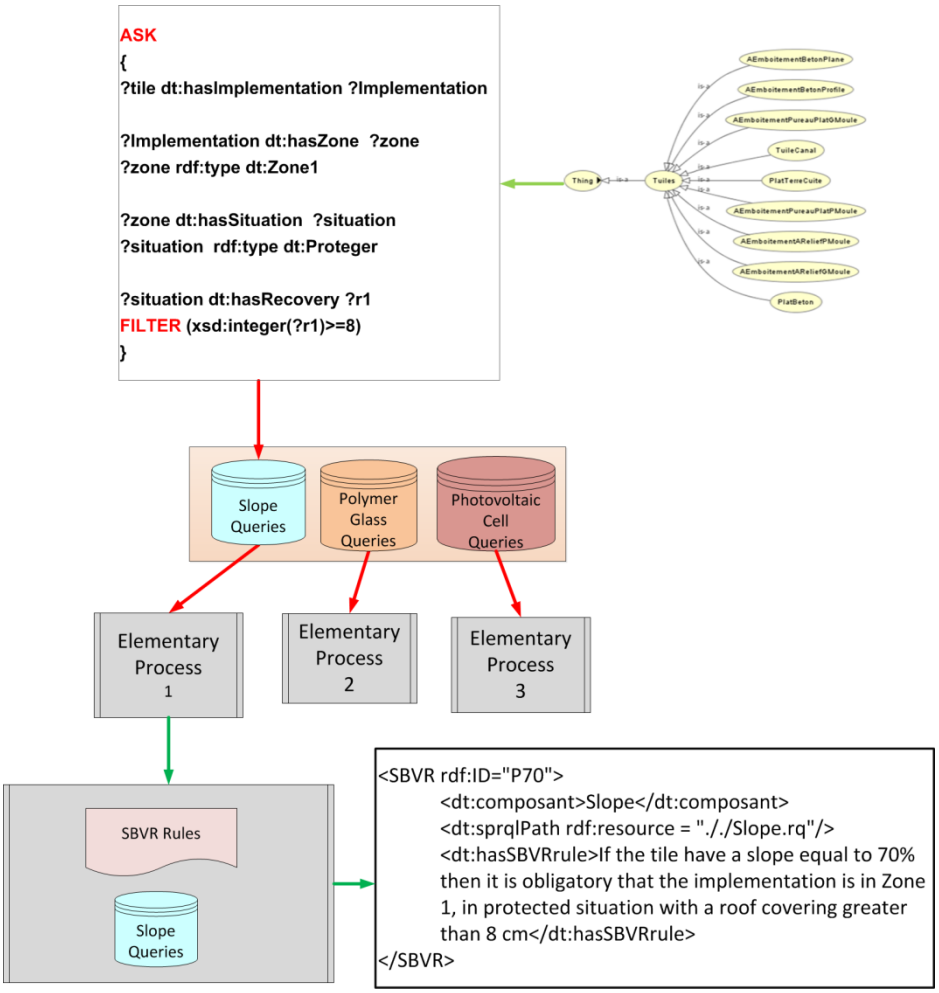
5.2. Elementary and Complex Processes

We distinguish between elementary and complex processes. A process is said to be elementary if it consists in the verification of the attributes of a component described in a TD which is denoted in our ontology by an atomic class. A process is said to be complex if it is associated to a component defined

in the ontology as a combination of sub-components. In that case, the process consists in the verification of the attributes of the components and the verification of those of its sub-components.

For instance, the validation process associated to a tile (Figure 12) is elementary: its RDF description calls for the execution of SPARQL queries testing its structural and dimensional criteria (the slope, the material, the form, *etc.*). In contrast, the validation process associated to PV glass polymer module is complex: its RDF description presented in (Figure 14) calls for the interpretation of the description of its sub-processes relative to the verification of its sub-components [Cadre, CellulePhotoV, FilmPolymere, VerreInterieur] (Section 3.3).

Figure 12. Elementary process of checking constraints.



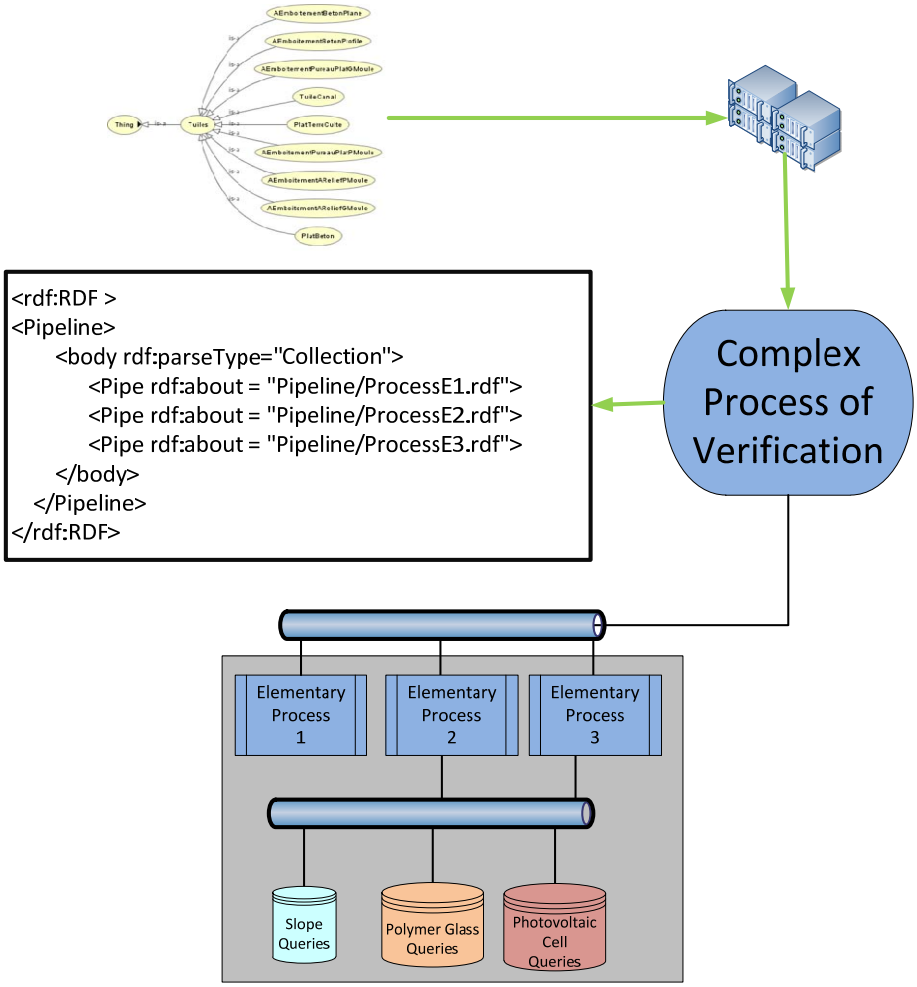
Such an RDF description of a complex process is automatically and dynamically generated, by using a SPARQL query template (Figure 13) querying the definition of the component to be validated in the OntoDT ontology.

Figure 13. A SPARQL query template for generating a complex process.

```
SELECT DISTINCT ?process where{
?x rdfs:subClassOf ?y
FILTER( ?x=dt:"Component-Name")
?y owl:onProperty ?p
FILTER( ?p=dt:aProcessus)
?y owl:hasValue ?process}
```

The interpretation and execution of such a process consists of the recursive interpretation of the description of its sub-processes. To be precise, in case of a complex process involving sub-processes, the process engine interprets the resource of type Pipe denoting a sub-process by loading its RDF description accessible at the URI and recursively interpreting it (see Figure 14).

Figure 14. Complex process of checking constraints.



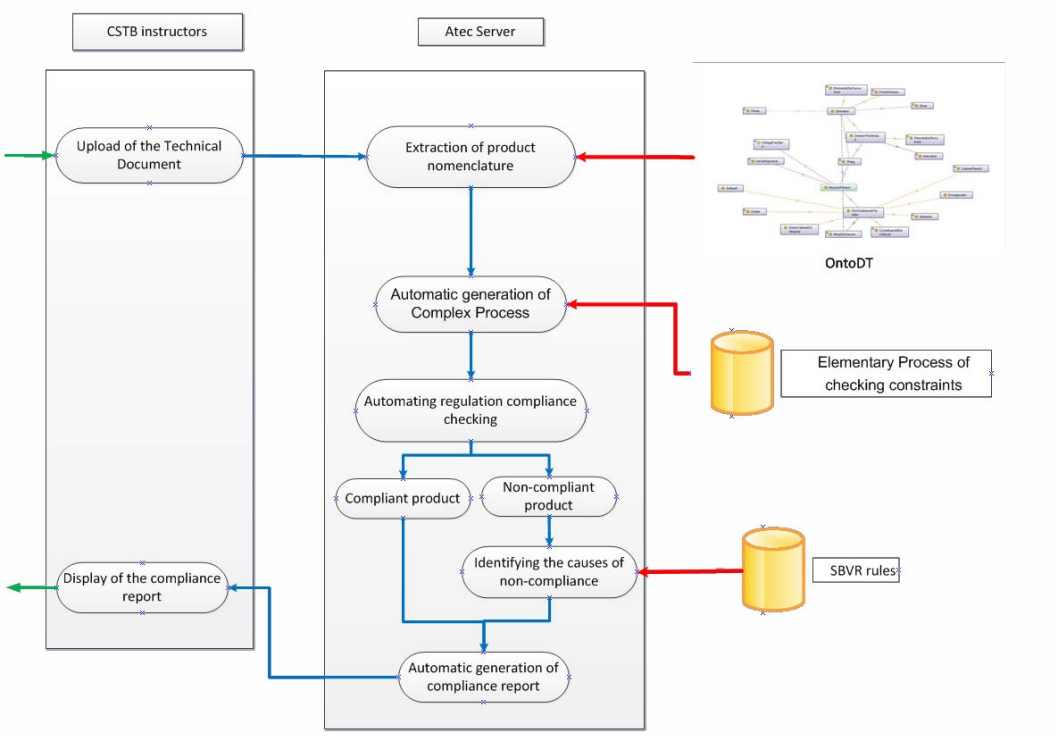
5.3. Automating Regulation Conformance Checking

To assist CSTB experts in writing technical advices, we have developed two tools (see Figure 15). The first one is dedicated to industrials and helps them in writing TDs. It uses the OntoDT ontology to

automatically build and present to the user the forms to fill in to describe the attributes of all the components of a given product. This tool outputs an RDF description of a product based on the information provided by the users.

This RDF description of the TD of a product comes as an input in a second tool we have developed to help CSTB experts in validating a TD against regulatory constraints. It first automatically builds an RDF representation of the verification process convenient for the product waiting for a technical advice. It uses the OntoDT ontology and the model described in the previous section to recursively build an RDF representation of the process, based on the definition of its components. Then it calls for the process engine developed for our process model and automatically produces to the user a conformity report. This report summarizes which of the queries representing technical constraints succeed when applied to the RDF description of a given TD, and which of them do not succeed, and therefore reveals a possible non conformity.

Figure 15. Overview of our assistant tool for writing technical advices.



6. State of the Art

Regulatory modeling is a vast research area. The authors of [17] have undertaken initial work on the structure of rules in decision tables. Decision trees were later applied in building industry, specifically in the design of steel buildings [18]. The SASE system [19], was developed to provide a complete hierarchical structure to classify families of related regulations or codes. A major study of these early approaches is provided in [20]. In addition, let us cite [21] who have developed the REGNET application to determine the applicability of building regulations in some given conditions, based on a question and answer interface.

Up to now, more recent work focuses on the rigorous extraction of requirements from regulations [22–24]. Authors presented a methodology for extracting stakeholder rights and obligations from regulations. Also, regulatory modeling was discussed under two different approaches. The first one aims to automatically analyze the rules and to confront the complexity of natural language [16,25]. In a second approach, regulatory constraints are directly written according to a normalizing model, with the help of domain experts, which facilitates their translation into formal models [26,27]. We propose a third approach, which takes into account the regulations written in natural language, offers a tool for writing TDs and automatically analyses the content of these documents and their conformance to regulation.

On the other hand, various efforts have been made to apply conformance rules to the representations of construction projects, using the structures of drawings specially coded (IFC) or textual descriptions [5,28,29]. When compared to these works, the originality of our approach lies in the combination of formal representations and SBVR rules from which they are derived to explain the rules themselves or the decision making. Our approach extends conformance checking with the explanation of the decision making.

7. Results and Limitations

We validated our approach on "real use case" at CSTB. We used TG which summarize the main requirements of seven Unified Technical Document (DTU) currently applied in France. We designed our rules database by extracting regulation constraints and transformed them into semi-formal language (SBVR) and formal language (SPARQL) using many steps. These transformations have been handled manually by domain experts. They were able to model 100% of TG constraints into SBVR.

Although the interpretation of experts helps in translating most of the constraints from text to SBVR, some of them remain non-transformable into SPARQL. These cases correspond to the "fuzzy" constraints that contain information defined in a qualitative way (e.g., "a short distance"), or that contain "common knowledge" [5] (e.g., "The recovery of the ridge tiles is in the opposite direction of the winds rain dominant"—Technical guides, "The tile roofs" page 75).

The current limitation of our technical guides is that they collect detailed execution data featuring a wide range of situations. These types of rules are called "Implementation rules", easily representable in SBVR, are the largest part of the rules that cannot be formalized in SPARQL. These rules represent a large proportion of regulatory constraints identified in the TGs (in this case 70%).

Example of implementation rules (Table 3):

Table 3. Transforming implementation rules into SBVR formulation.

Implementation rules	SBVR formulation
«In the case of Canal tile, it typically runs in a mortar or flashing bardelis embedded and sealed in the wall.» *Technical guides "The tile roofs" page76* [30].	**If tile canal then it is obligatory that it *runs* in a mortar or flashing bardelis embedded and *sealed* in the wall.**

As final results, with the help of CSTB experts, we identified about 177 SPARQL queries and 177 elementary processes. This result is about 30% of the SBVR rule modeled.

8. Conclusions

In this paper, we have presented an approach and a tool to assist in compiling regulation and technical documents in the construction industry and partially automate regulation conformance checking. We propose a domain ontology, OntoDT, representing concepts involved in the description of technical documents and regulations. We combine SBVR and semantic web languages representing a controlled vocabulary and formalize regulatory constraints extracted from the Practical Guides edited by CSTB. These two complementary representations both are based on the OntoDT ontology and are associated in RDF annotations. The SBVR-based representation of regulations presented to the users, reduces ambiguity. RDF and SPARQL based formalizations enable automating regulation conformance checking. Finally, we propose a process model to organize regulations extracted from Technical Guides. A regulation conformance checking process is associated to each basic component involved in a technical document and a whole checking process is automatically built in based on the RDF description of the technical document and the component definitions in the OntoDT ontology.

We have developed a tool for representing technical documents and another one for their regulation conformance checking. They have been evaluated by CSTB instructors who have validated both the granularity of the information required and the OntoDT ontology. Our regulation conformance checking tool is a component of an assistance tool in writing technical advices, which we have developed for CSTB instructors and which is not detailed within this paper. Our models of regulation, technical documents and conformance checking process have been evaluated through the evaluation of this tool. CSTB instructors have validated the messages provided by our tool.

A major perspective of our work is the identification of regulations which cannot be completely formalized and their inclusion through SBVR representations in a semi-automatic conformance checking process.

References

1. Web Ontology Language (OWL). Available online: http://www.w3.org/TR/owl-ref/ (accessed on 28 August 2012).
2. Semantics of Business Vocabulary and Business Rules (SBVR). Available online: http://www.omg.org/spec/SBVR/ (accessed on 28 August 2012).
3. SPARQL Protocol and RDF Query Language (SPARQL). Available online: http://www.w3.org/TR/rdf-sparql-protocol/ (accessed on 28 August 2012).
4. Resource Description Framework (RDF). Available online: http://www.w3.org/RDF/ (accessed on 28 August 2012).
5. Yurchyshyna, A. Modélisation Du Contrôle de Conformité en Construction: Une Approche Ontologique (in French). Ph.D. Dissertation, University of Nice Sophia Antipolis, Sophia Antipolis, France, 2009.
6. Gehre, A.; Katranuschkov, P.; Stankovski, V.; Scherer, R.J. Towards semantic interoperability in virtual organizations. In *Proceedings of the 22nd Conference on Information Technology in Construction*, Dresden, Germany, 19–21 July 2005.

7. Bus, N.; Fies, B.; Bourdeau, M.; Charvier, M.; Labedens, R. *Reef Sémantique, Diffusion et Application des Textes Technico-Réglementaires* (in French); Accompagnement des pouvoirs publics dans la rédaction des textes officiels: CSTB, Sophia Antipolis, France, 2009.
8. SERVLET, JAVA Programming Language Class. Available online: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html (accessed on 28 August 2012).
9. Corby, O.; Faron-Zucker, C. The KGRAM abstract machine for knowledge graph querying. *Web Intell.* **2010**, *1*, 338–341.
10. Lau, G.T.; Law, K.H.; Wiederhold, G. Analysis government regulations using structural and domain information. *Computer* **2005**, *38*, 70–76.
11. Bolioli, A.; Dini, L.; Mercatali, P.; Romano, F. For the automated mark-up of Italian legislative texts in XML. In *Proceedings of Jurix—15th Annual International Conference on Legal Knowledge and Information Systems*, London, UK, 23–24 May 2002.
12. Turk, Z.; Katranuschkov, P.; Scherer, R.J.; Cerovsek, T. The tools and services integration platform of the ISTforCE project. In *Proceedings of 3rd International Conference on Concurrent Engineering in Construction*, Berkeley, CA, USA, 1–3 July 2002.
13. Object Management Group (OMG). *Business Semantics of Business Rules*; br/2003-06-03; OMG: Needham, MA, USA, 2003. Available online: http://www.omg.org/cgi-bin/doc?br/03-06-03 (accessed on 28 August 2012).
14. OMG. *Semantics of Business Vocabulary and Business Rules (SBVR)*; OMG: Needham, MA, USA, 2006. Available online: http://www.omg.org/spec/SBVR/1.0/PDF/ (accessed on 28 August 2012).
15. Chapin, D.; Baisley, D.E.; Hall, H. Semantics of business vocabulary & business rules (SBVR). In *Proceedings of W3C Workshop on Rule Languages for Interoperability*, Washington, DC, USA, 27–28 April 2005.
16. Martínez-Fernández, J.L.; González, J.C. A preliminary approach to the automatic extraction of business rules from unrestricted text in the banking industry. In *Proceedings of 13th International Conference on Applications of Natural Language to Information Systems*, London, UK, 26–28 June 2008.
17. Fenves, S.J. Tabular decision logic for structural design. *J. Struct. Div.* **1966**, *92*, 473–490.
18. Nyman, D.J.; Fenves, S.J.; Wright, R.N. *Restructuring Study of the Aisc Specification*; Civil Engineering Standards, Structural Research Series 393; University of Illinois at Urbana-Champaign: Champaign, IL, USA, 1973; pp. 473–490.
19. Fenves, S.J.; Wright, R.N.; Stahl, F.I.; Reed, K.A. *Introduction to SASE: Standards Analysis, Synthesis, and Expression*; National Technical Information Service: Gaithersburg, MD, USA, 1987; pp. 473–490.
20. Fenves, S.J.; Garrett, J.H.; Reed, K.A. Computer representations of design standards and building codes: U.S. perspective. *Int. J. Constr. Inform. Technol.* **1995**, *3*, 13–34.
21. Kerrigan, S.; Law, K.H. Logic-based regulation compliance-assistance. In *Proceedings of 9th International Conference on Artificial Intelligence and Law*, Edinburgh, Scotland, 24–28 June, 2003; pp. 126–135.
22. Breaux, T.D.; Vail, M.W.; Antón, A.I. Towards Compliance: Extracting rights and obligations to align requirements with regulations. In *Proceedings of the 14th IEEE International Conference on Requirements Engineering*, Minneapolis, MN, USA, 11–15 September 2006; pp. 49–58.

23. Breaux, T.D.; Antón, A.I. Mining rule semantics to understand legislative compliance. In *Proceedings of the ACM Workshop Privacy in the Electronic Society*, Alexandria, VA, USA, 7 November 2005; pp. 51–54.
24. Breaux, T.D.; Antón, A.I. Analyzing regulatory rules for privacy and security requirements. *IEEE Trans. Softw. Eng.* **2008**, *34*, 5–20.
25. Dinesh, N.; Joshi, A.; Lee, I.; Sokolsky, O. Reasoning about conditions and exceptions to laws in regulatory conformance checking. *Deontic Log. Comput. Sci.* **2008**, *5076*, 110–124.
26. Reeder, R.W.; Karat, C.M.; Karat, J.; Brodie, C. Usability challenges in security and privacy policy-authoring interfaces. *Interact* **2007**, *4663*, 141–155.
27. Nazarenko, A.; Guisse, A.; Levy, F.; Omrane, N.; Szulman, S. Integrating written policies in business rule management systems. *Rule-Based Reason. Program. Appl.* **2011**, *6826*, 99–113.
28. Pauwels, P.; van deursen, D.; Verstraeten, R.; de Roo, J.; de Meyer, R.; van de Walle, R.; van Campenhout, J.A. Semantic rule checking environment for building performance checking. *Autom. Constr.* **2011**, 20, 506–518.
29. Eastman, C.; Lee, J.M.; Jeong, Y.S.; Lee, J.K. Automatic rule-based checking of building designs. *Autom. Constr.* **2009**, *18*, 1011–1033.
30. Technical Guides (The Tile Roofs). Available online: http://boutique.cstb.fr/media/ensavoirplus/145/ext_Couverture.pdf (accessed on 28 August 2012).

© 2012 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).