Automatic Detection and Semantic Formalisation of Business Rules

Cheikh Kacfah Emani[1,2]

[1] CSTB, 290 route des Lucioles, BP 209, 06904 Sophia Antipolis, France
[2] Université Lyon 1, LIRIS, CNRS, UMR5205, 69622, France

Abstract. Experts in construction engineering are overwhelmed by regulatory texts. It is a heavy task to go through these texts and get an unambiguous list of requirements they contain. Moreover, with regard to the number of texts and the diversity of their writers, we cannot neglect the possibility of getting inconsistencies. Finally, these requirements are to be put close to digital representation of buildings to detect potential non-conformities. This paper examines these problems and envisions solutions to help experts. We thus envisage to automate detection and extraction of business rules in regulatory texts. Next, we propose to formalise identified requirements as `SPARQL` queries. These queries will serve for conformity checking on `OWL`-representation of buildings. Moreover, we plan to leverage these queries to detect inconsistencies in regulatory texts.

Keywords: #eswcphd2014Emani.

1 Introduction

Several pieces of text govern the field of construction engineering. In addition to the literary freestyle, these texts can be understood in various ways. In the digital era, it is desirable to automate the reformulation of these Business Rules (BR) with more simple phrases. Further, these regulations have to be followed by buildings. Knowing that they are designed more and more by means of computers, we can plan an automatic verification of building mock-up's conformity. To this end, conformity requirements need to be represented in a way that allows for checking them against digital representations of buildings. Challenges proposed by this problem written in a single sentence are multiple. First we must navigate within regulatory texts and help experts in construction engineering (henceforth denoted by "building experts" or simply "experts") to detect and extract minimal phrases that act as requirements. This task requires a set of NLP functionalities, which we denote $\mathcal{F}$, like: ($\mathcal{F}_1$) identification, within a sentence containing multiple requirements, of all the words which constitute a given rule, ($\mathcal{F}_2$) detection of scope of applications and contexts of rules, ($\mathcal{F}_3$) stemming, ($\mathcal{F}_4$) paraphrases, linguistic co-references, ellipsis and mentions to other pieces of texts, ($\mathcal{F}_5$) order of premises and corresponding conclusions, ($\mathcal{F}_6$) highlighting of "implicit parameters" (e.g: "The length must be sufficient" means that the length must be *greater than a certain value*), etc. Results of this extraction must be put in a form easily verifiable by building experts. Next these requirements must be written in a computer understandable language. Similarly the building should be represented in a vocabulary *compatible* with conformity policies. Finally, non conform elements in buildings must be signalled to building experts. With its set of standards and easy to handle tools, the Semantic Web seems to be a promising source of support for the formalisation part of our problem. Practically, many recent works have successfully taken advantage of it [6],[23]. As we will see later, many tasks described in these PhD works are manual. "Automation" (as much as possible) is therefore the leitmotiv of this work. Our goal is discussed through the following agenda. Initially, the state of the art on detecting, extracting, rephrasing and formalising of conformity requirements is introduced (Sect. 2). Next, the problem is presented through an example and we point out possible contributions and primary insights of solution (Sect. 3). Finally, we expose a plan to validate our future results (Sect. 4).

2 State of the Art

Two fields of study are the most relevant to our work: *(i)* detecting and rephrasing rules and *(ii)* automatically formalising them.

Detecting and Reformulating Business Rules. Some proposal have been made to ease writing of policies [8],[20]. In our case we assume that they have already been written and we wish to disambiguate them by cleaning them to keep only their core formulation. Indeed, a BR is supposed to be *atomic, compact, well-formed, unambiguous* and *built upon domain-vocabulary* [9]. This aim is considered to be impossible [16] or in more optimistic words complex.

Facing the necessary functionalities listed in the introduction, a good compromise for this task is to assist experts [2], [7], [15,16], [19]. In [19], a detailed but manual methodology is proposed to identify, extract and formalise rules. Such hand-done execution gives the impression that lot of functionalities (e.g ($\mathcal{F}_3$) to ($\mathcal{F}_6$)) are not implemented. More easy to delegate to machines, Breaux and Anton [7] take advantage of goal mining [1, chap. 4], to tackle security policy management. They rephrase goal statements in "restricted natural language statements" (*RNLS*). RNLS are obtained by splitting each goal into sentences with exactly one actor and action, and must contain the essence of the original goal. Although leading to an expressive and query-able model, it is applied manually. Consequently, few of the constraints listed above are broached. More complete (except ($\mathcal{F}_2$)), a deeply detailed architecture and processes to achieve extraction and refinement of rules is given in [16]. There, with the help of a terminology extraction tool, `TERMINAE` [3,4], the user builds a domain ontology from texts. Next, an "index" is output [10]. This index links each piece of text to its concepts and to its rules. Similarly to the recommendations of Bouzidi [6], these rules are written w.r.t SBVR-SE (Semantics of Business Vocabulary and

Business Rules - Structured English). The editing of these rules is done manually. During this process, the user is helped by the highlighting of domain terms in the rule [15].
A fully automatic tool for BR generation is presented in [5]. There, Bajwa et al. take as input a text written in natural language and output *rephrased* SBVR rules. Their tool has a fairly good accuracy (87.33%), however it does not handle constraints solved by ($\mathcal{F}_4$) - non appearance of business terms in regulatory texts - and ($\mathcal{F}_6$) - requirement submitted to experts' subjective appreciation.

Formalisation of Rules. The rephrasing of rules or policies in simple terms as discussed earlier paves the way for more ambitious aims. In [7], the authors use a kind of table ($Activity < actor, action, object >$) to store information. Obviously, such representation is not expressive enough for all type of requirements. It is thus languages of the Semantic Web which obtain favours when modelling become more complex [6], [21,22,23]. Indeed, they are suitable because of their expressiveness, interoperability, standardization. Yurchyshyna [23] was the first to propose to represent building requirements as `SPARQL` queries. By doing it, she intended to query the `OWL`-representation of buildings to detect non-conform elements. This successful proposition has been followed by Bouzidi [6]. But in these two theses this process is done by hand. The automation of such transformation is addressed within many papers [21,22]. These works are relative to `SPARQL`-isation of questions w.r.t an `RDF`-knowledge base. They want to find and rank possible answers to natural language questions. To address this issue, they propose a two-steps approach. First, `SPARQL`-template(s) of the question is(are) output. Secondly, each pattern is instantiated through an entity linking resolution process. Entity linking aims at identifying the most suited entity in an unambiguous knowledge base to which refers a given string. They use a domain independent lexicon to formalise recurrent terms like *who, the most, at least, etc.* A domain-dependent lexicon is also required to represent the general behaviour of terms. For instance, if they are generally *property* or *object*, their possible types, or *domain* and *range*. This domain-dependent lexicon needs to be built almost manually. Unfortunately this construction is a tedious task.

3 Problem Statement and Contributions

3.1 Use Case Description

The regulatory text taken as example here is a (non-official) translation of the second item of the second paragraph of the subsection 6.1 of the article 6 found in the first chapter of the "Arrêté" mentioned in [13]. This text is about vertical interior circulation of the common parts of residential buildings.

```
Safety in use: At the top of the stairs, flooring must allow waking
alertness at a distance of 0.50 m from the first step through a
visual and tactile contrast.
The first and last steps must be provided with a riser with
```

```
a minimum height of 0.10 m, visually contrasting with walking.
The nosings must meet the following requirements:
- Being visually contrasting with respect to other stairs;
- Be non-skid;
- Present no excessive overhang relative to the riser.
The staircase must have a lighting device that meets the
requirements set out in Article 10.
```

From this piece of text, the following “conformity” BR were extracted and validated by building experts:

Table 1. BR extracted from [13] and their SBVR rephrasing w.r.t IFC vocabulary (**Colour code**: concept - *verb* - **value** - SBVR keyword)

Fully include in the text	**Rephrased and including words from business vocabulary**
(1) At the top of the stairs, flooring must allow waking alertness at a distance of 0.50 m from the first step	*At the top* of stairs , the length *of* the slab *must be* at least **50 cm**
(2)-(3) The first/last step must be provided with a riser with a minimum height of 0.10 m	The height *of* the riser *of* the first/last step *must be* at least **0.10 m**
(4) The nosings must be visually contrasting with respect to the rest of the stairs	The contrast *between* the nosings and the steps *must be* at least **<slot>**
(5) The nosings must be non-skid	The nosings *must* *have* non-skid surface
(6) The nosing must present no excessive overhang relative to the riser	The nosing length *must be* at least **<slot>**

It is important to notice that, in practice, we have to resolve the co-reference implied by the phrase *“requirements set out in Article 10”*. In addition, a quick look at Tab. 1 helps us to illustrate the list of constraints given in Sect. 2.

We can now put our SBVR-written rules in processable form. As suggested in [23] we choose `SPARQL` language[1]:

1. Rule (1):

```
SELECT ?stair ?length
WHERE {?stair rdf:type ifc:IfcStair.
OPTIONAL{
     ?stair ifc:IfcRelAggregates ?slab.
     ?slab rdf:type ifc:IfcSlab.
     ?slab ifc:PredefinedType ifc:LANDING.
     ?slab ifc:LENGTH ?length.
     FILTER (?length >=
       "500"^^xsd:positiveInteger)}.
     FILTER (! bound (?length))}
```

2. Rules (2) and (3):

```
SELECT ?stair ?height
```

[1] Declarations of common prefixes like `owl`, `xsd`, `rdf`, `rdfs` and the prefix of our ontology `ifc` are omitted for readability purposes.

```
WHERE {?stair rdf:type ifc:IfcStair.
OPTIONAL{
     ?stair ifc:IfcRelAggregates ?stairFlight.
     ?stairFlight rdf:type ifc:IfcStairFlight.
     ?stairFlight ifc:RiserHeight ?height.
     FILTER (?height >=
       "100"^^xsd:positiveInteger)}.
     FILTER (! bound (?height))}
```

3. Rule (5):

```
SELECT ?stair ?isNonSkid
WHERE { ?stair rdf:type ifc:IfcStair.
OPTIONAL{
     ?stair ifc:HasCoverings ?buildingElement.
     ?buildingElement ifc:RelatedCoverings
      ?covering.
     ?covering ifc:PredefineType ifc:FLOORING.
     ?covering ifc:HasNonSkidSurface ?isNonSkid.
     FILTER (?isNonSkid = "true"^^xsd:boolean)}.
     FILTER (! bound (?isNonSkid))}
```

4. Rule (6):

```
SELECT ?stair ?length
WHERE { ?stair rdf:type ifc:IfcStair.
OPTIONAL{
     ?stair ifc:IfcRelAggregates ?stairFlight.
     ?stairFlight rdf:type ifc:IfcStairFlight.
     ?stairFlight ifc:NosingLength ?length.
     FILTER (?length <=
       "10"^^xsd:positiveInteger)}.
     FILTER (! bound (?length))}
```

The above queries assume that:

– Like ontoCC [23, sect. 5.2], we suppose an ontology, `ifc`, which matches IFC entities, object and data properties [17] and so on. Moreover, it is an IFC-represented "object", which therefore instantiate `ifc`, which is queried.
– In addition, as expressed in [23], we are looking for non conform characteristics of our "object". Non conformity includes a value *mismatch* or its *lack*. We thus use the function `! bound()` to check if variables we are interested in, have been bound or not.

When we look closely at these queries, we can formulate a set of challenges brought up by this problem:

– Conversion from BR to `SPARQL` is not straightforward: BR are not in a simple *<subject, verb (property), object>* form. Such a form could have made us foreseen this conversion as a mapping task: relate each element of the triple to a single element of `ifc`.
– Decoding of some recurrent words as operator has to be done (e.g: *at least* in queries 1 and 2) or in some cases, aggregation functions like `COUNT`, `MAX`, `MIN`, etc.
– Detecting implicit operands as asking. For instance in case for operand of binary operators (e.g: $isNonSkid = true$ in query 3) or in case of empty slot (e.g: minimal length (10) in query 4).
– Handling of units of measurement. In our queries, lengths are expressed in millimetres.
– Going further than *syntactic* manipulations. Indeed, some URIs in the query do not appear (even if we look at the rule through stemming, compliant editing distance, synonyms, etc.) in the original rule. It is the case of properties like `ifc:IfcRelAggregates`, `ifc:PredefinedType` and `ifc:LANDING` in query 1.

– In the current state of IFC (IFC 4), it is not always possible to express a BR as a query (e.g: rule (4)).

In practice, our set of queries are stored. They will be triggered according to user specifications. For instance a user may want to verify only accessibility (stairs, lifts, doors, etc.) or lightening, etc. It means that `SPARQL` queries have to be annotated. Similarly some requirements are compulsory and others are just recommendations. All such metadata have to be added to queries and taken into account during the checking process. More details are provided in [23, chap. 6]. But there the organisation of the base of conformity queries is manual.

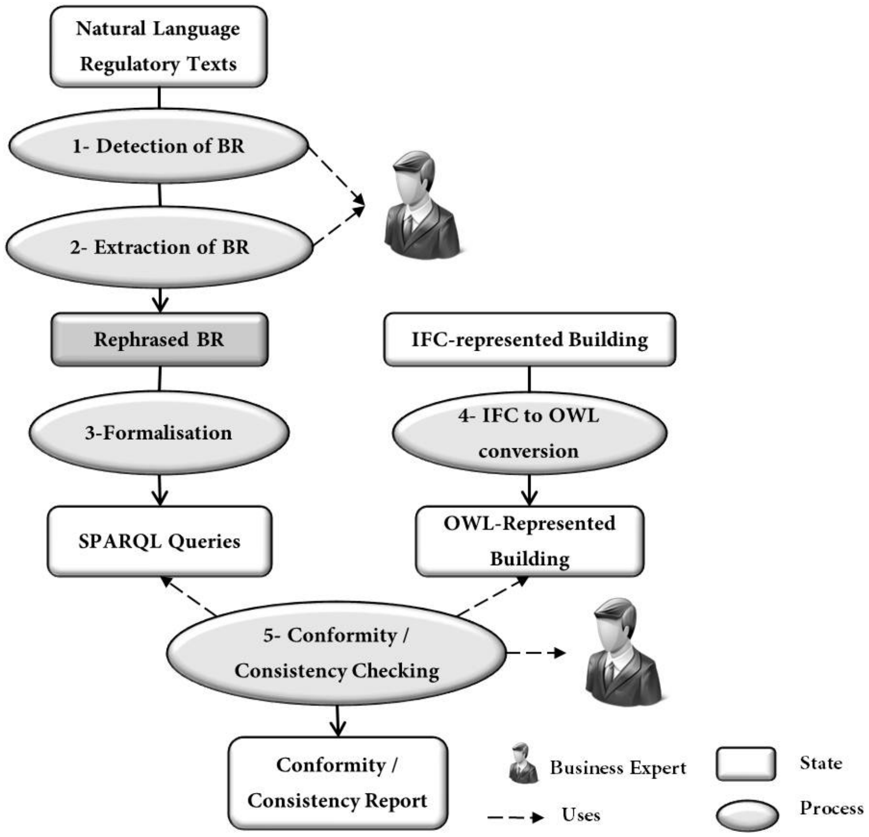

Fig. 1. The whole conformity checking process

3.2 Overview of Pursued Solutions and Approach

We have presented many sides of the problem of automation of conformity checking[2]. Some state-of-the-art methodologies and tools have been exposed. Now we present extensions, ameliorations and novel approaches.

[Steps 1 and 2] Rules Detection and Rephrasing. This is the entry point of our supply-chain (Fig. 1). As we have seen, the automation of this task is very complicated and a good trade-off is a semi-automation. First we must flatten lists and solve co-references. Here, the co-reference resolution process must disambiguate coordinators for establishing semantic relations between pieces of texts, and must handle co-reference within a paragraph (out of scope of a single sentence) as in [14]. Next, based on identification of key terms (from vocabulary or common words like minimal, at-least, etc.) and various heuristics, a proposition of possible rules should be made. These heuristics could be made on the structure (pattern) of sentences and the presence of certain words. Since it will happen to output non-sense (phrases as) rules or to have undetected rules, we intend to ask user for correction and in background seamlessly trigger a learning process. Heuristics are usually based on deep observations. Consequently, they fail due to more or less scarce exceptions. We can thus envisage a learning process designed for such context like *Ripple Down Rules (RDR)* [11]. Indeed, RDR are suitable when we want to add few exceptional cases to a set of conditional expressions and help to limit the number of conditions to be updated.

[Step 3] Automatic Formalisation of BR. Tens of languages could be used for this task. But since `SPARQL` has been successfully used [6], [23] it has our favour. But it is not the only reason. `SPARQL` is a standard and is popular in the Semantic Web community. Moreover, it is used to query `RDF` graphs and very expressive languages have been build on top of `RDF`. We principally have in mind `OWL`. It means that providing a `SPARQL` version of conformity requirements does not restrict expressiveness of digital representation of building. In addition, promising methods for full automation of translation of natural language questions to `SPARQL` have been developed [21,22]. Consequently, we can leverage methodology exposed by Unger, Lehmann and their colleagues. For our task, an extension of their domain-independent dictionary is needed (*minimal*, *maximal*, *more or less*, etc.). More we must propose an algorithm which gets around the use of the domain-dependent lexicon since it is heavy to build.
The main goal of this algorithm is to deduce triple patterns, between concept and predicate appearing in a simple phrase, by taking only the ontology of the domain (which covers this phrase) as input. We see in [21,22] that this relation is usually straight. For example in the sentence *Who produced the most films?*,

[2] We have taken here only a piece of a regulatory text to illustrate our work. For our final solution we must leverage a representative sample of regulatory texts. This representativeness is mainly about NLP challenges brought up by these texts. Moreover, the quality of the sample must be validated by the experts.

the relation between `produced` and `films` is direct. In our use case, we see in query 3 that starting at `ifc:IfcStair` and reaching `ifc:HasNonSkidSurface` leads us to pass through three intermediate properties.

Also, as mentioned earlier in this document, some operands are implicit. It is the case of the "value" *not excessive overhang* converted in *at least <slot>* in the sixth rule. This slot has been filled by the value **10**. Such value varies from one expert to another. Our job there consists to detect these cases and create slots which filling will be done at execution. This goal could be achieved using `SPIN` [12].

Let us mention that we do not focus our attention on the "step 4". It was addressed in Yurchyshyna's thesis [23].

[Step 5] Inconsistency Detection. When introducing this work, we have underlined the fact that the process described here is triggered by the volume of regulatory texts and the diversity of writers. This situation can lead to redundancies and more problematically to inconsistencies. Thus, assuming the formalisation of requirements, we could help building experts to correct texts. If requirements are represented by `SPARQL` queries, the challenge is to identify incompatible triple patterns in `SPARQL` graph patterns. Hypothetically, we have thousands or even millions of them. Therefore, we cannot plan to do pairwise comparisons. A possible method to reduce them is to index our queries. Since inconsistencies can only be found through `FILTER` clauses (excluding those about the binding), triple patterns they contain are good candidate for our keys. If we take for example `SPARQL` query number 1, the filter is applied on a *length.* When we scan the query, we see that the length is attached to *slabs.* So this requirement could be indexed by the "(length, slab)" entry. Consequently, it will be compared only with potential queries with the same entry to see if their filters are compatible. Another possible method to avoid useless comparisons is to cluster queries like in [18].

To Summarize. The whole process, from consuming raw regulatory texts to the delivering of a conformity or consistency report, is depicted by Fig. 1.

This figure clearly emphasises the five (or six) points of our work. First, with help of building experts, requirements are identified and extracted. Next we aim to automatically formalize these requirements. Simultaneously, the building is brought from IFC to `OWL`. Results of all these processes allow us to think about conformity checking. During this process we can prompt an expert for signalling non-formalised BR or to ask for implicit value as explained in Sect. 3.2. In addition, as reported on the figure, we can verify consistency of texts through their `SPARQL` representation. Unlike the conformity checking process, the consistency one does not need the building representation. In conformity report we have the concerned piece of regulation text and the precise reason of non conformity [23]. In consistency report, we present conflictual rules to the expert.

4 Evaluation Plan

Many theses have walked through the successive steps we have described. But their works have multiple manual tasks. Nevertheless, results they obtain constitute a good way to validate our work. So, our detection and extraction processes can be applied on the different corpus used by Bouzidi [6] and Yurchyshyna [23]. Since the rules they have extracted have been validated by experts, these rules will be used to evaluate the precision and the recall of our detection and extraction steps. A similar approach is planned about formalisation. Also important, we will focus on the usability of our future tool.

5 Conclusion and Perspectives

This paper presents the work envisioned for our PhD thesis. Its essence can be expressed with the word automation. At the end of our work, we plan to provide algorithms to automate detection and extraction of rules from regulatory texts. Next, we intend to re-write automatically these requirements by `SPARQL` queries (and corresponding annotations) aligned with IFC vocabulary. This task implies be able to identify all implicit functions, triple patterns and filter parameters without human help. This formalisation should be easy to apply in any domain, assuming we have its ontology. Finally we plan to propose a framework to detect incompatibility between a set of `SPARQL` queries supposed to describe coherent requirements.

References

1. Anton, A.I.: Goal Identification and Refinement in the Specification of Information Systems. Ph.D. thesis, Georgia Institute of Technology (June 1997)
2. Anton, A.I., Earp, J.B., He, Q., Stufflebeam, W., Bolchini, D., Jensen, C.: Financial privacy policies and the need for standardization. IEEE Security & Privacy 2(2), 36–45 (2004)
3. Aussenac-Gilles, N., Biebow, B., Szulman, S.: Revisiting ontology design: A methodology based on corpus analysis. In: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management, EKAW 2000, pp. 172–188. Springer, London (2000)
4. Aussenac-Gilles, N., Despres, S., Szulman, S.: The TERMINAE method and platform for ontology engineering from texts. In: Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap Between Text and Knowledge, pp. 199–223. IOS Press, Amsterdam (2008)
5. Bajwa, I.S., Lee, M.G., Bordbar, B.: SBVR business rules generation from natural language specification. In: AAAI Spring Symposium: AI for Business Agility, pp. 2–8. AIII (2011)

6. Bouzidi, K.R.: Aide à la création et à l'exploitation de réglementations basée sur les modèles et techniques du Web sémantique. Ph.D. thesis, Université Nice Sophia Antipolis (September 2011)
7. Breaux, T.D., Anton, A.I.: Analyzing goal semantics for rights, permissions, and obligations. In: Proceedings of the 13th IEEE International Conference on Requirements Engineering, RE 2005, pp. 177–188. IEEE Computer Society, Washington, DC (2005)
8. Brodie, C.A., Karat, C.M., Karat, J.: An empirical study of natural language parsing of privacy policy rules using the sparcle policy workbench. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, pp. 8–19. ACM, New York (2006)
9. Graham, I.: Business rules management and service oriented architecture: A pattern language. John Wiley & Sons, Chichester (2006)
10. Guissé, A., Lévy, F., Nazarenko, A.: Un moteur sémantique pour explorer des textes réglementaires. In: Mille, A. (ed.) Actes des 22èmes Journées Francophones d'Ingénierie des Connaissances, pp. 451–458. Publibook (2011)
11. Kim, M.H.: Ripple-Down Rules based Open Information Extraction for the Web Documents. Ph.D. thesis, School of Computer Science and Engineering, The University of New South Wales, Australia (April 2012)
12. Knublauch, K., Idehen, K., Hendler, J.A.: SPARQL inferencing notation (SPIN), `http://spinrdf.org/`
13. Lecomte, A., Trégoat, J.: Arrêté du 1er aout 2006 fixant les dispositions prises pour l'application des articles R. 111-18 à R. 111-18-7 du code de la construction et de l'habitation relatives à l'accessibilité aux personnes handicapées des bâtiments d'habitation collectifs et des maisons individuelles lors de leur construction. Journal Officiel 195(13), 111–118 (2006)
14. Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., Jurafsky, D.: Deterministic coreference resolution based on entity-centric, precision-ranked rules. Computational Linguistics 39(4), 885–916 (2013)
15. Levy, F., Guisse, A., Nazarenko, A., Omrane, N., Szulman, S.: An environment for the joint management of written policies and business rules. In: Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, vol. 2, pp. 142–149. IEEE Computer Society, Washington, DC (2010)
16. Lévy, F., Nazarenko, A.: Formalization of natural language regulations through SBVR structured english. In: Morgenstern, L., Stefaneas, P., Lévy, F., Wyner, A., Paschke, A. (eds.) RuleML 2013. LNCS, vol. 8035, pp. 19–33. Springer, Heidelberg (2013)
17. Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Richter, S., Chipman, T., Weise, M., Wix, J.: IFC4 official release, `http://www.buildingsmart-tech.org/ifc/IFC4/final/html/`
18. Lorey, J., Naumann, F.: Detecting SPARQL query templates for data prefetching. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 124–139. Springer, Heidelberg (2013)
19. Maxwell, J.C., Anton, A.I.: Developing production rule models to aid in acquiring requirements from legal texts. In: Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE 2009, pp. 101–110. IEEE Computer Society, Washington, DC (2009)

20. Reeder, R.W., Karat, C.-M., Karat, J., Brodie, C.: Usability challenges in security and privacy policy-authoring interfaces. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4663, pp. 141–155. Springer, Heidelberg (2007)
21. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.C., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 639–648. ACM, New York (2012)
22. Unger, C., Cimiano, P.: Pythia: Compositional meaning construction for ontology-based question answering on the semantic web. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 153–160. Springer, Heidelberg (2011)
23. Yurchyshyna, A.: Modélisation du contrôle de conformité: une approche ontologique. Ph.D. thesis, Université Nice Sophia Antipolis (February 2009)