Semantic Norm Modeling and Business Process Regulatory Compliance using LegalRuleML

Guido Governatori¹, Mustafa Hashmi¹, Ho-Pun Lam¹, Serena Villata² and Monica Palmirani³

1 Data61, CSIRO, NICTA
2 CNRS – Laboratoire d’Informatique, Signaux et Systèmes de Sophia-Antipolis
3 CIRSFID, University of Bologna

Abstract. Legal documents are the source of norms, guidelines, and rules that often feed into different Web applications. In this scenario, it is important to have a sufficiently expressive conceptual model of the various heterogeneous aspects of such norms, guidelines, and legal knowledge in general, in order to support the further development and deployment of successful applications. In this paper, we report on the modelling and interlinking of the legal rules from an industry code, namely the Australian Telecommunications Consumer Protections Code. We investigate how to exploit Semantic Web technologies and languages like LegalRuleML to model such document through rules as well as its dynamics, in order to support humans in analysis tasks over such legal texts. Moreover, we show how this semantic annotations are used to empower a business process regulatory compliance system. We finally discuss the open challenges raised by applying a semantic approach to the legal domain.

1 Introduction

Business Process Management (BPM) is a set of methodologies to capture, model and control in an integrated way all those activities taking place in an environment defining the enterprise [7]. Companies are subject to regulations, and resulting not to be compliant to such regulations affects the added-value of the business processes of the company, and may even be the cause of judiciary pursuits. The scope of norms is to regulate the behaviour of their subjects and to define what is legal and what is illegal. In BPM, checking the compliance of a business process with respect to the available regulations means to identify whether a process violates or not a set of norms. Intuitively, we say that a process is compliant with a set of norms, if it does not breach such a normative system. For an effective and successfully application of ICT (Information and Communication Technology) techniques to the problem of ensuring the business processes are compliant with the relevant regulations we need two components: (i) a conceptually sound formal representation of a business process, and (ii) a conceptually sound formalism to model and to reason with the regulations. However, the task of modeling legal information, e.g., a code, a bill, an act, in a satisfactory way is far from being straightforward, and requires a powerful language to capture the whole semantics of the normative system and its dynamics, i.e., a code is amended to a new version. For these reasons, current solutions to BPM are not fully satisfiable for companies.
In this paper, we present a framework for semantic business process regulatory compliance checking. Relying on the semantics of LegalRuleML model [2, 3], we discuss and analyse different but comparable ways to model the semantics of norms as well as their dynamics (e.g., new versions of a certain code are proposed). Moreover, we show how this semantic modeling phase, coupled with a semantic annotation, always based on LegalRuleML, of the tasks of the process, can be exploited to address and improve regulatory compliance checking. We experiment our approach on two versions of the Australian Telecommunications Consumer Protections Code.

In particular, we present how Semantic Web languages and technologies can be exploited to answer companies’ needs about regulatory compliance checking. The evaluation we addressed together with domain experts from the industry partner shows that our solution actually allows to overcome some of the drawbacks of standard non-semantic approaches to compliance checking usually adopted in companies.

The reminder of the paper is the following. In Section 2, we describe the context in which our approach has been conceived, answering the needs expressed by an industry partner, and we highlight how these needs can be casted in BPM research. Section 3 describes how we model norms using LegalRuleML, and how we exploit the semantics of LegalRuleML to perform semantic regulatory compliance checking using the Regorous system. In Section 4, we report on the results of the evaluation of the system, and we discuss the main insights we inferred from this experience. Finally, we compare the related literature with the proposed approach in Section 5, and draw some conclusions.

2 The Context

Regulatory compliance is the set of activities an enterprise does to ensure that its core business does not violate relevant regulations, in the jurisdictions in which the business is situated, governing the (industry) sectors where the enterprise operates.

The activities an organisation does to achieve its business objectives can be understood as business processes, and consequently they can be represented by business process models. On the other hand, a normative document (e.g., a code, a bill, an act) can be understood as a set of clauses, and these clauses can be represented in an appropriate formal language. Based on this, Governatori and colleagues [15] proposed that business process compliance is a relationship between the formal representation of a process model and the formal representation of a relevant regulation. For more details about the different strategies to gain compliance, we refer the reader to [26].

Two issues have to be taken into account when regulatory compliance is targeted: on the one side, we have the extreme interest of companies in continuous improvements of the quality of services, and on the other side, we have to account for changing legislations and compliance requirements. To obviate these problems, the idea of compliance-by-design has been introduced [27]. The key aspect of the compliance-by-design methodology is to supplement business process models with additional information to ensure that a business process is compliant with relevant normative frameworks before the deployment of the process itself.

We provide now the vary basics of business process modelling, for an extensive presentation we refer the reader to [7]. A business process model is a self-contained, temporal and logical order in which a set of activities are executed to achieve a business goal. Typically a process model describes what needs to be done and when (control flow), who is going to do what (resources), and on what it is working on (data). Many different formalisms (Petri-Nets, Process algebras, ...) and notations (BPMN, YAWL, EPC, ...) have been proposed to represent business process models. Besides the difference in notations, business process languages typically contain the following minimal set of elements:

- tasks, i.e., a (complex) business activity, and
- connectors, i.e., the relationships among tasks to be executed, like sequence, and-join, and-split, (x)or-join, (x)or-split.

The combination of tasks and connectors defines the possible ways in which a process can be executed. Where a possible execution, called trace, is a sequence of tasks respecting the order given by the connectors.

Compliance is not only about the tasks that an organisation has to perform to achieve its business goals, but it is concerned also on their effects (i.e., how the activities in the tasks change the environment in which they operate), and the artefacts produced by the tasks (for example, the data resulting from executing a task or modified by the task). To check whether a business process is compliant with a relevant regulation, we need an annotated business process model and the formal representation of the regulation. The annotations are attached to the tasks of the process, and it can be used to record the data, resources and other information related to the single tasks in a process.

In this paper, we report the results of a project in cooperation with an industry partner subject to the Australian Telecommunications Customers Protection Code. The industry partner did not have formalized business processes. Thus, we worked with domain experts from the industry partner (who had not been previously exposed to BPM technology, but who were familiar with the industry code) to draw process models for the activities covered by the code. The main objective of the project is to exploit semantic technologies to empower the compliance-by-design methodology. More precisely, our objectives are as follows:

- Modeling the regulatory code as well as its dynamics using a (machine-readable) semantic framework such that differences and connections between two versions of the code (namely, 2012 and 2016) can be automatically identified;
- Capturing the tasks, their effects and the artifacts resulting from them by means of a semantic annotation of the tasks in a process model;
- Extend the architecture of the Regorous Process Designer (from now on simply Regorous), a business process compliance checker based on the business process compliance methodology proposed by Governatori and Sadiq [17], to account for the semantic annotation.

The reminder of the paper describes the proposed architecture, and the evaluation we carried out together with domain experts from the industry partner.

5 The name of the industry partner as well as the processes cannot be provided due to commercial reasons.
3 The Framework

In this section, we first introduce the main insights of LegalRuleML as well as the model (Section 3.1). Second, we describe how LegalRuleML can be exploited to model the Australian Telecommunications Consumer Protections Code, and how to connect and underline differences and similarities in the content of the two versions (2012 and 2016) of the Code (Section 3.2). Finally, we describe the semantic annotations in LegalRuleML we associated to the tasks of the processes, and how they are used to address semantic regulatory compliance checking (Section 3.3).

3.1 LegalRuleML: motivations and model

LegalRuleML is an effort to create a standard for the representation of norms. The aim of this section is to give a concise overview of LegalRuleML, for the background, motivation and detailed presentation of the language and example of its use the reader can consult [24, 2, 3]. A key tenet in LegalRuleML is that rules are an appropriate representation of norms [28]. Accordingly, LegalRuleML builds on the experience of RuleML to provide a rule representation language and, at the same time, LegalRuleML extends RuleML following the principles and guidelines proposed in [12] for rule languages for legal reasoning. In particular, LegalRuleML offers facilities to model different types of rules (i.e., constitutive and prescriptive rules), deontic effects (e.g., obligations, prohibitions, permissions), and rules with different strengths (i.e., strict, defeasible and defeasible rules). In addition, it has features to capture the metadata of norms and normative documents (e.g., jurisdiction, authorities, validity times), and it has mechanisms to allow for the so called legal isomorphism [4] principle that establishes the connection between a legal source or a norm, and the corresponding formal representation.

A LegalRuleML document has three components (Fig. 1): metadata, statements and contexts.

![LegalRuleML Document](image)

Fig. 1: LegalRuleML document structure.

The metadata part is meant to contain the legal sources of the norms modelled by the documents, and information about the (legal) temporal properties (or characteristics) of the norms. LegalRuleML is just about to enter its public review phase.
of the sources and the document itself, the jurisdiction where the norms are valid, and eventually details describing the authorities, authors, . . . for the legal sources and document.

The statement part contains the formal representation of the norms in form of rules or other expressions in the language. In LegalRuleML, statements are classified as depicted in Figure 2.

![Fig. 2: Types of statements in LegalRuleML.](image)

Normative statements follow the well known distinction of constitutive statements (rules) and prescriptive statements (rules). Constitutive rules are used to provide the definitions of the terms used in the document. For example, Chapter 2 of the Code provides the definitions of the terms used in the rest of Code. Notice, that as common in legal reasoning, terms are defined defeasibly, thus the definition gives the base conditions that can be further extend or are subject to exceptions. For example, the Code defines “Complaint” as:

an expression of dissatisfaction made to a Supplier in relation to its Telecommunications Products or the complaints handling process itself, where a response or Resolution is explicitly or implicitly expected by the Consumer.

An initial call to a provider to request a service or information or to request support is not necessarily a Complaint. An initial call to report a fault or service difficulty is not a Complaint. However, if a Customer advises that they want this initial call treated as a Complaint, the Supplier will also treat this initial call as a Complaint.

If a Supplier is uncertain, a Supplier must ask a Customer if they wish to make a Complaint and must rely on the Customer’s response.

Given the nature of the definition of the terms in legal reasoning, the definition of the terms can be captured by defeasible rules. In other terms, the constitutive rules provide the internal (defeasible) ontology used by the LegalRuleML document.

Prescriptive statements are the rules that determine what are the obligations, prohibitions, permissions (and other deontic effects) the parties affected by the norms are subject to; in addition, such rules provide the conditions under which the deontic effects are in force.

Factual statements are meant to capture facts that are relevant in given cases or for a LegalRuleML document: for example, a factual statement can be used to specify that a particular manifestation of a norm (e.g., Section 8.4.1 of the the 2012 version of the
Code) is the same as another manifestation of the norm (i.e., Section 8.3.1 of the 2016 version of the Code).

Given that the norms are represented by defeasible rules, and that two defeasible rules can be in conflict, override statements are used to solve the conflicts by specifying that in case two rules in conflict fire at the same time, the stronger rule prevails over the weaker rule (notice, however, that each override statement just provides the relative strength of two rules).

Finally, violation and reparation statements offer convenient ways to formalise the penalties that can potentially apply for breaches of many norms, and information about what norms or rules are compensated by which penalty.

3.2 Modelling the code and its dynamics

The Telecommunication Consumer Protections Code is the Australian industry code for the telecommunication industry and mandates that every operator in the section has to provide annual compliance statements with the code. The code was enacted in September 2013, and it entered in force from April 2013. In 2015, the code was revised, and a new version enacted, then some amendments entered in force at the beginning of 2016. In this paper, we consider the 2012 and 2016 versions of the Code, and how to model them in LegalRuleML.

The first option to model the two versions of the Code is to have two separate LegalRuleML documents: one for the 2012 version of the Code, and the second for the 2016 version. When we compare the two versions of the Code, we realise that while there are differences, the vast majority of the definitions and prescriptions are the same in the two versions. Thus, the option just described will result in a large duplication of the rules used to represent the code.

The second option is to use the LegalRuleML’s ability to link rules with their legal sources using the facilities provided by the <lrml:Context> and <lrml:Association> elements. For this option the first step is to create a metadata part where we have the information about the two versions of the code. Thus we have two LegalSource elements (with their elements. For this option, the first step is to create a metadata part where we have the information about the two versions of the Code. Thus, we have two LegalSource elements (with their metadata information about enactment time, in force time, jurisdiction and so on).

<lrml:LegalSources key="ls1">
  <lrml:LegalSource key="tcpc2012"
  <lrml:LegalSource key="tcpc2016"
</lrml:LegalSources>

The second step is to have a single statement section where we collate all statements irrespective of whether they are for the 2012 or the 2016 version of the Code. Thus, we obtain something like the following schema (see Section 3.3 for how to actually model the norms as LegalRuleML statements).
For the final step we create two lrml:Context elements, one for each version of the Code. Then, for each context, we have to specify the associations between each section in the Code and the rules corresponding to the section.

In the example above, Section X.Y is the same in both versions of the Code, and it is modelled by the same rule tcpc_ps1. However, Section X.W is modelled by rule tcpc_ps2 in the 2012 version and rule tcpc_ps3 in the 2016 version.

The option we just described does not require to duplicate the rules; however, we still have to list all the associations for all the provisions in the Code, and the corresponding
rules. A more compact alternative would be to have a single association for each context with a single source (the entire version of the Code) and multiple targets (the rules corresponding to that version). Notice that this option has the drawback to loose the semantic information about the relationship between the sections of the Code and corresponding rules (syntactically, the correspondence could be regained by establishing a schema for labelling the key of the rules).

3.3 Business process regulatory compliance

The set of traces $T$ of a given business process describes the behavior of the process insofar as it provides a description of all possible ways in which the process can be correctly executed. To check the semantic regulatory compliance of a process, we consider it as the set of its traces. The set of norms could vary from a particular regulation, to a specific statutory act, to a set of best practices, a standard, or simply a policy internal to an organisation or a combination of these types of prescriptive documents.

In this section, we introduce the architecture of the Regorous Process Designer, a business process regulatory compliance checker [18], and how we extended it by enriching both the representation of norms and the business process tasks with the LegalRuleML semantic model. Starting from the norms modeled in LegalRuleML presented in Section 3.2, we need now to add such semantic annotations in LegalRuleML to the tasks of the process, using them to record the data, resources and other information related to the single tasks in a process.

For the formal representation of the regulation, Regorous use FCL [13, 16]. FCL is a simple, efficient, flexible rule-based logic, obtained from the combination of defeasible logic (for the efficient treatment of exceptions which are quite common in normative reasoning), and a deontic logic of violations. In FCL, a norm is represented by a rule of the kind

\[ a_1, \ldots, a_n \Rightarrow c \]

where $a_1, \ldots, a_n$ are the conditions of applicability of the rule and $c$ is the normative effect of the rule. FCL distinguishes two normative effects: the first is that of introducing a definition for a new term. For example, the following rule from the Australian Telecommunications Consumer Protections Code (2012) specifies that if a Customer requests information about a Complaint, then it is deemed a consumer complaint activity.

\[ \text{complaint}, \text{request\_information} \Rightarrow \text{consumer\_complaint\_activity} \]

The second normative effect is that of triggering obligations and other deontic notions. For obligations and permissions, we use the following notations:

- \([P]p\): $p$ is permitted;
- \([O\!P]p\): there is a punctual obligation for $p$;
- \([O\!M]p\): there is a maintenance obligation for $p$;
- \([O\!A\!P\!P]p\): there is an achievement preemptive and perdurant obligation for $p$;
- \([O\!A\!P\!N\!P]p\): there is an achievement preemptive and non-perdurant obligation for $p$;
- \([O\!A\!N\!P\!P]p\): there is an achievement non-preemptive and perdurant obligation for $p$;
– [OANPNP]p: there is an achievement non preemptive and non-perdurant obligation for p.
– [OM]¬p: p is prohibited;

Rules involving obligations and permissions are much more complex. Let us consider the following example from the Code: The supplier must implement, operate and comply with a Complaint handling process that is transparent, including prohibiting a Supplier from cancelling a Consumer’s Telecommunications Service only because, being unable to Resolve a Complaint with their Supplier, that Consumer pursued their options for external dispute resolution. This rule is translated into FCL as follows:

\[
\neg \text{resolution, request\_information, external\_dispute\_resolution} \Rightarrow [\text{OM}]\neg \text{terminate\_service.}
\]

It is worth mentioning that FCL is agnostic about the nature of the literals it uses. They can represent tasks, i.e., activities executed in a process, or propositions representing state variables. For full description of FCL and its feature see [13, 16].

The first extension we have applied to the Regorous architecture was the ability to generate and parse both xml and LegalRuleML versions of the regulations, i.e., the Code. For instance, the above rule is translated using the LegalRuleML semantic model as follows:

```xml
<lrml:PrescriptiveStatement key="ps_tcpc_8_1_1_a_x_E">
  <ruleml:Rule key="tcpc_8_1_1_a_x_E">
    <lrml:Paraphrase>The supplier must implement, operate and comply with a Complaint handling process that is transparent, including prohibiting a Supplier from canceling a Consumer’s Telecommunications Service only because, being unable to Resolve a Complaint with their Supplier, that Consumer pursued their options for external dispute resolution. This rule is translated into FCL as follows:
    <lrml:Description>
      \[
      \neg \text{resolution, request\_information, external\_dispute\_resolution} \Rightarrow [\text{OM}]\neg \text{terminate\_service.}
      \]
    </lrml:Description>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```
As shown in the previous section, enriching the regulatory compliance system with a semantic representation of the regulations the processes have to be checked against presents many advantages, i.e., a more insightful and precise representation of the semantics of the norms, and the possibility to keep track of the regulations’ dynamics.

However, compliance is not just about the tasks to be executed in a process but also on what the tasks do, the way they change the data and the state of the artefacts related to the process, and the resources linked to it. Accordingly, process models must be enriched with such information [27]. For this reason, we decided to enrich process models with semantic annotations using the LegalRuleML model. Each task in a process model is then attached to a set of semantic annotations in LegalRuleML, representing the effects of the tasks. The approach can be used to model business process data compliance [19].

An example of task annotated using the LegalRuleML semantics is the following: suppose that the complaint handling process of a telco contains a task called "Record Complaint". The Code (Section 8.5 of the 2012 version, and Section 8.5 of the new version) specifies what information should be recorded for a complaint. Thus, the task "Record Compliant" indicates that such an activity is to be performed once a compliant as been verified as such, but, the process alone does not specify what data is recorded. Thus, such process model must be extended with the appropriate information. Note that it is beyond the scope of this paper to study how the annotations are generated, i.e., manually based on domain experts knowledge of the process or by examining database schemas associated to the task or programming script executed by the task [19]. Specifically, for the task “Record Complaint” the following literals (from the literals defined for the LegalRuleML document) are recorded as annotation for the task:

```xml
<taskEffects elementId="usertask15">
  <ruleml:Atom>
    <ruleml:Rel>record special circumstances</ruleml:Rel>
  </ruleml:Atom>
  <ruleml:Atom>
    <ruleml:Rel>record complaint issue</ruleml:Rel>
  </ruleml:Atom>
  <ruleml:Atom>
    <ruleml:Rel>record resolution sought</ruleml:Rel>
  </ruleml:Atom>
  <ruleml:Atom>
    <ruleml:Rel>record due date</ruleml:Rel>
  </ruleml:Atom>
  <ruleml:Atom>
    <ruleml:Rel>record complaint cause</ruleml:Rel>
  </ruleml:Atom>
</taskEffects>
```

Figure 3 depicts the architecture of Regorous extended with the semantics of the LegalRuleML model, where the red boxes identify the extended components. Note that the automated extraction of the rules (e.g., in FCL) from the natural language legal text is out of the scope of the present paper. In this paper, we start from the assumption that the rules are provided at the beginning of the compliance checking phase, being them
extracted automatically from texts or generated by human experts. Given an annotated process and the formalisation of the relevant regulation in LegalRuleML as we shown above, we can use the algorithm proposed in [16] to determine whether the annotated process model is compliant. Shortly, the process runs as follows:

- Generate an execution trace of the process.
- Traverse the trace:
  - for each task in the trace, cumulate the effects of the task using an update semantics (i.e., if an effect in the current task conflicts with previous annotation, update using the effects of the current tasks).
  - use the set of cumulated effects to determine which obligations enter into force at the current tasks, by calling a reasoner.
  - add the obligations obtained from the previous step to the set of obligations carried over from the previous task.
  - determine which obligations have been fulfilled, violated, or are pending; and if there are violated obligation check whether they have been compensated.
- repeat for all traces.

In conclusion, a process is evaluated as compliant if and only if all traces are compliant (all obligations have been fulfilled or if violated they have been compensated), or it is evaluated as weakly compliant if there is at least one trace that is compliant.
4 Evaluation

In this section, we first present our evaluation of the proposed semantic approach (Section 4.1), and then we discuss the lessons learned with this experience together with the industry partner (Section 4.2).

4.1 Results

The approach proposed in this paper has been evaluated in a six week pilot project in collaboration with an industry partner, and the regulator. For the evaluation, Chapter 8 of the Code on complaint handling was selected. A legal knowledge engineer from our group manually mapped Chapter 8 of the 2012 version of code in LegalRuleML, and XSLT transformations are used to translate the LegalRuleML representation in FCL as used by Regorous. The mapping of Chapter 8 took approximately 2 weeks. The chapter contains approximately 100 commas, in addition to approximately 120 terms given in the Definitions and Interpretation section of the Code (Chapter 2). The mapping resulted in 176 LegalRuleML normative statements, containing 223 distinct rulelm1:Atoms, and 7 lrlm:overrides statements. Of the 176 normative statements, 33 were lrlm:ConstitutiveStatements used to capture definitions of terms used in the remaining rules. Mapping the section of the Code required all features of FCL. The regulator examined the mapping, and they deemed it to be an suitable interpretation of the Code.

For the second phase of the evaluation, we had a series of 1-day workshops with the industry partner. The industry partner did not have formalised business processes. Thus, we worked with domain experts from the industry partner (who had not been previously exposed to BPM technology, but who were familiar with the industry code) to draw process models for the activities covered by the Code. The evaluation was carried out in two steps. In the first part, we modelled the processes as they were. It took two workshops to teach them how to model business process, and to jointly model their existing processes. The third 1-day workshop was dedicated to add the semantic annotation to the business process. The domain expert were able to complete the task in one afternoon after they were instructed on how to do in the morning.

Regorous was able to identify several areas where the existing processes were not compliant with the new code. In some cases the industry partner was already aware of some of the areas requiring modifications of the existing processes. However, some of the compliance issues discovered by the tools were novel to the business analysts and were identified as genuine non-compliance issues that need to be resolved. In the final part of the experiment, the existing processes were modified to comply with the code based on the issues identified in the first phase. In addition, a few new business process models required by the new code were designed. As result, we generated and annotated 6 process models. 5 of the 6 models are limited in size and they can be checked for compliance in seconds. The largest process contains 41 tasks, 12 decision points, xor splits, (11 binary, 1 ternary). The shortest path in the model has 6 tasks, while the longest path consists of 33 tasks (with 2 loops), and the longest path without loop is 22 task long. The time taken to verify compliance for this process amounts approximately to 40 seconds on a
MacBook Pro 2.2GHz Intel Core i7 processor with 8GB of RAM (limited to 4GB in Eclipse).

4.2 Lessons learned and future work

The results presented in the previous section demonstrate the effectiveness of the semantically enriched business process regulatory compliance checking mechanism we proposed. Besides these considerations, some further positive and negative insight emerged during the evaluation we conducted with the industry partner:

**Positive feedback:** First, we discovered, together with the domain experts of the industry partner, that exploiting a semantic model such as LegalRuleML allows us to embed much more information in the rules representing the regulatory code. This enhancement in the precision of the legal provisions leads to an enhancement of the regulatory checking phase, as we compared two more fine-grained representations of the Code and of the tasks, respectively. Second, the semantics of LegalRuleML allows to account for the evolution over time of the regulations to be compliant with. This has the advantage of tracking when a change occurred, and what is the context to be used depending if the compliance is verified against the new or the old version of the Code. This is also a valuable benefit of the adoption of this semantic model with respect to simple rule-based formats.

**Negative feedback:** Even if the semantic model allows to provide a more faithful representation of the legal document, it is not straightforward to understand the semantic model of LegalRuleML and how it works. It required some time to the domain experts of the industry partner to understand how to match the rules present in the Code they were aware of, and the LegalRuleML semantics. As future work, we need to develop a graphical interface to interact with such a complex model so that examples of rules in natural language from existing regulations are provided and translated into LegalRuleML so that the experts are supported in their modeling phase. In summary, the result of our experience with the industry partner showed us that users are much better in using semantically enriched documents rather than in creating them. Finally, the showstopper is that the extraction of the rules from the legal documents is time consuming and there is a huge need to support the translation of such legal documents from natural language to their XML counterpart. This is another line we will address as future work.

5 Related Work

The problem of providing a machine-readable semantic representation of legal knowledge has been addressed in different domains, leading to the definition of various ontologies targeting different legal contexts, like for instance, the Creative Commons (CC) ontology\(^7\) and the Open Digital Rights Language (ODRL) for rights expressions\(^8\). Among others, there are the Functional Ontology for Law (FU) [29] about normative knowledge,

---

\(^7\) https://creativecommons.org/ns

\(^8\) https://www.w3.org/ns/odrl/2/ODRL21
world knowledge, and responsibility knowledge, the Frame-Based Ontology of Law (FBO) [21] about norms, acts and concepts descriptions, the IKF-IF-LEX Ontology for Norm Comparison [9] about agents, instuitive norms, instrumental provisions, regulative norms and norm dynamics, the LKIF-Core Ontology including the OWL ontology of fundamental legal concepts [25]. Moreover, Boella and colleagues have proposed an ontological model of norms based on the concept of agency [5], and they have addressed the issue of representing interpretation of terms besides the definitions occurring in the EU directives [1]. Finally, Gangemi and colleagues showed how legal ontologies can be exploited to create newly designed legal decision support systems [11], and he proposed design patterns for legal ontologies [10]. However, all these works differ from LegalRuleML for what concerns (i) the use of rules to account for the specifics of the legal domain, and (ii) the use of a legal reasoning level on top of the ontological layer of the Semantic Web stack. Moreover, LegalRuleML allows to specify in different ways how legal documents evolve, and to keep track of these evolutions and connect them to each other, as we exploited in this paper.

To our knowledge, there is no other approach addressing the problem of a semantic business process regulatory compliance: in this work, we not only exploited Semantic Web technologies and languages to propose different modelling techniques to represent the legal information contained in the legal documents and their dynamics, i.e., the Telecommunications Code (2012 and 2016 versions), but we empowered a business process compliance system with a semantic annotation of the rules and the processes. An approach to semantic business process compliance management has been proposed by El Kharbili and colleagues [20, 6]. We share the idea of making use of the advantages of semantic technologies for compliance management. However, the two approaches are different. First of all, we are interested in regulatory compliance, and not in business process management in general, which means that we need to exploit the powerful semantics of the LegalRuleML framework to convey the semantics of the rules extracted from the legal documents. We do not need to design a business policy and business rule ontology, as in [20, 6]. Second, the proposed architecture considers also the dynamics of the legal documents to be checked the compliance with, by proposing alternative modeling solutions. Finally, we annotate the tasks included in the processes with the semantic of LegalRuleML, so that automated compliance checking is done at semantic level. Besides Regorous a few other compliance prototypes have been proposed. Here we consider some representative ones: MoBuCom [23], Compass [8] and SeaFlows [22]. However, none of them exploits Semantic Web languages and technologies, and they are not compliant with the guidelines set up in [12] for rule languages for the representation of legal knowledge and legal reasoning. In addition, such approaches have severe limitations in modelling legal reasoning, since they do not provide a conceptually sound model of legal reasoning [14].

6 Conclusions

In this paper, we have presented a semantic approach to business process regulatory compliance checking. Regulatory compliance checking is a major challenge for companies
and institutes, and being supported by automated techniques in such a verification phase results in a valuable gain of time and money. We have reported here our experience in applying Semantic Web technologies and languages to this challenging task in the context of a project with an industry partner. Accounting for the complexity and the required precision in modelling norms and regulations, and in checking whether a certain process and its related tasks are actually compliant with the normative system they are subject to, we propose a semantic approach based on the LegalRuleML semantic model. Our evaluation shows that our system is able to capture the semantics of the Code and to model its dynamics in a satisfactory way, and to efficiently check the compliance of processes with respect to this reference Code. The lessons learned during this project will guide our future work, that includes also the evaluation of the Regorous semantic system with larger processes, and applying this methodology with other kinds of regulations in order to make Regorous more flexible to the needs of the companies adopting it.

References