CoReL: Policy-Based and Model-Driven Regulatory Compliance Management

Marwane El Kharbili*, Qin Ma*†, Pierre Kelsen*† and Elke Pulvermueller‡
*Laboratory for Advanced Software Systems
†Interdisciplinary Centre for Security Reliability and Trust
University of Luxembourg, Luxembourg
‡Department of Mathematics & Computer Science, Institute of Computer Science
University of Osnabrueck, Germany

***Abstract*—Regulatory compliance management is now widely recognized as one of the main challenges still to be efficiently dealt with in information systems. In the discipline of business process management in particular, compliance is considered as an important driver of the efficiency, reliability and market value of companies. It consists of ensuring that enterprise systems behave according to some guidance provided in the form of regulations. This paper gives a definition of the research problem of regulatory compliance. We show why we expect a formal policy-based and model-driven approach to provide significant advantages in allowing enterprises to flexibly manage decision-making related to regulatory compliance. For this purpose, we contribute CoReL, a domain-specific modeling language for representing compliance requirements that has a graphical concrete syntax. Informal semantics of CoReL are introduced and its use is illustrated on an example. CoReL allows to leverage business process compliance modeling and checking, enhancing it with regard to, among other dimensions, user-friendliness, genericity, and traceability.**

***Keywords*-Regulatory Compliance; Policy; Business Processes; Domain Specific Language.**

I. Introduction

Compliance management is the discipline of ensuring that enterprises behave according to certain constraints. Usually, when speaking of regulatory compliance, these constraints on enterprise behavior are expressed as regulations, i.e., documents such as laws, contracts, norms and standards or any other form of internal or external guidance (e.g., internal risk management directive documents). In distributed systems, compliance problems are often studied as security problems (e.g., UMLSec[1]), or, more rarely, as quality problems (e.g., QoS [2]). In academia, compliance is acknowledged to be a broad and challenging topic, encompassing modeling, checking and monitoring compliance [3], [4], [5], [6]. The number of published papers on the topic has been constantly increasing since the year 2001 [4]. This trend shows just how much there is at stake in providing a genuine understanding of the problem of compliance as well as a comprehensive solution.

One of the main drivers behind the increasing interest in compliance are the financial scandals that have been happening in the financial world since the beginning of the year 2000 in North America (Enron, WorldCom, etc.) as well as in Europe (Societe Generale, Parmalat, etc.). In the rest of the paper, we will use the following definition of regulatory compliance management (RCM):

Definition I.1. *[Regulatory Compliance Management] RCM is the problem of **ensuring** that enterprises (data, processes, organization, etc.) are **structured** and **behave** in accordance with the regulations that apply. In the opposite case we say that a company is **violating** a regulation.* □

The main challenge that academia and industry have been facing is to come up with techniques and tools to efficiently deal with RCM. We concretely mean here the ability to perform automatic and regular compliance checks, and to generate compliance reports. In enterprises, automation is a critical aspect of cost optimization, and business processes play a central role in this surge towards automation. In particular, enterprises seek to pro-actively manage compliance and control the occurrence of violations, by explicitly specifying accepted violations as well as preferred ways to react accordingly.

This paper proposes a model-driven solution to the RCM problem. We start in Section II by positioning our work with regard to the main relevant streams of related work, and uncover several shortcomings formulated as success criteria we want to address. In Section III, we introduce our approach to compliance management. We then proceed in Section IV to the presentation of the abstract syntax model of our Compliance Requirement Language (CoReL), the core contribution of our solution to the RCM problem. Section V shows how the modeling of a simple regulation can be done using CoReL , and finally, Section VI discusses the fulfillment by CoReL of the elicited success criteria, while Section VII details planned future work and presents concluding remarks.

II. Existing Approaches to Business Process Compliance Management

A. Rule- Vs. Policy-Based

Research on compliance management for business processes has brought up a wide variety of techniques for the modeling and the verification of compliance. For space reasons, we only cover the most relevant ones for our work here. We divide them into two categories. **Rule-based**

approaches use web rule languages (e.g., SWRL[1], RIF[2], RuleML[3], WSML[4]), modeling constraint languages such as OCL[7], languages coming from the business rule engine vendors or related bodies (e.g., PRR[5]) and modal languages (temporal such as CTL and LTL [8], deontic, etc.). **Policy-based** approaches make the management of the typically high number of rules to be followed by enterprises easier by grouping, structuring and prioritizing rules into policies. Also, policy languages may include deontic modalities (e.g., obligation) and model policy powers (e.g. delegation), while providing analysis and enforcement functionalities. Examples of such approaches are Ponder[9], KAOS[10], Rei[11], XACML[6], and Rewerse[12].

In both rule-based and policy-based approaches, process models are annotated with either rules or policies that can be checked and be evaluated to True or False. Moreover, business processes possess a dynamic nature (execution), thus requiring specific types of logic to be used in the definition of a rule or policy language for compliance. Furthermore, non-monotonic reasoning is required to deal with defaults and violations [13].

Compliance management for business processes consists of different tasks: modeling, verification at design-time, enforcement and monitoring at run-time as well as post-execution analysis.

B. Related Approaches

One of the most relevant approaches for our work is based on the Formal Contract Language (FCL) [14] language developed in recent years by Governatori on the basis of defeasible logic. FCL can be used for the expression of CoReL rules, because FCL allows to represent and reason about deontic modalities and violations. Next to FCL, the work by the authors of [15] is based on the definition of pre- and post-conditions defined for each process task which are checked against predefined clauses modeling the regulation. On the other hand, Namiri [16] concentrates on the modeling of internal controls as business rules. In [17], constraints are modeled and attached to process definitions, and are verified on the process models depending on the changes undergone by the process models. All of them explicitly annotate processes with rules that are checked. We follow the same approach to define the semantics of enrichment of process models with policies. Liu et al. [18] provide a comprehensive approach for static compliance checking of BPEL[7] processes with a graphical language for modeling LTL-based compliance requirements and using a model checker. Another class of works uses formal representations of process modeling notations (e.g., Petri Nets [19], or the REO coordination language [8], which supports a powerful language for temporal logic). Temporal rules are verified by using model checking techniques on these formal representations.

C. Identified Shortcomings

Here we list some issues that are not addressed or only partially addressed by existing approaches:

1) Business User Orientation: Business users are the real target users of a compliance management framework. Most often, it seems reasonable to assume that logical formalisms are not directly usable by business users due to their (sometimes overwhelming) complexity for non logicians or non-computer scientists.

2) Modularity: It is one thing to create adequate formal rules allowing to check compliance, it is another to efficiently manage them. How easy is it to deal with ever growing rule/policy sets? How easy is it to reuse or modify them? Can compliance knowledge already existing in the form of rules or policies be put to profit? A modular language allowing to model compliance requirements in a "lego" fashion by putting together different pieces of compliance knowledge should make compliance management easier.

3) Flexibility: Dealing with compliance by using customized software or code-implementation allows handling a single aspect of compliance and for a specific set of systems or applications. This makes compliance implementation hardly able to cope with (continuous) change in regulations.

4) Multiple Business Domains: Compliance management can be required at different layers of enterprise management. It can be needed at early requirement elicitation, business process, web service or system architecture level. A truly generic approach that is orthogonal to these layers would yield significant advantages in ease of compliance management through a single language and a uniform method.

5) Multiple Regulation Types: The use of current approaches is intended for one specific compliance problem and they are difficult to reuse for solving other types of compliance problems [20]. Some solutions deal with the problem of segregation of duty, the second-set-of-eyes problem [16], access control (most often RBAC-based [21], [22] ADD Pretschner's OSL), law [23] and other security or quality properties.

6) Multiple Logical Formalisms: The challenge of compliance checking requires the use of powerful formal verification techniques that allow to reason over a rich variety of logical aspects of the formal modeling of compliance requirements. Due to the richness of regulatory documents and the wide variety of compliance requirements they pose on enterprises, a single formalism (e.g., a rule language) is most often not sufficient to deal with the totality of a

[1] http://www.w3.org/Submission/SWRL/

[2] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

[3] http://ruleml.org/

[4] http://www.w3.org/Submission/WSML

[5] www.omg.org/spec/PRR

[6] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[7] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[8] REO Coordination Language http://reo.project.cwi.nl

regulation. For example, solely relying on a business rule language will not allow capturing temporal constraints on business processes, and vice versa, a temporal logic will not allow capturing the whole range of structural constraints on a process.

7) Traceability: Implementations of compliance requirements may be scattered over many logically expressed elements. It thus becomes harder to keep track of which requirements were implemented, why, and how they relate to the implementations of other requirments.

8) Holisticness: It is not straightforward to understand how the existing approaches can deal with the whole enterprise model that includes not only business processes but also other relevant aspects such as enterprise organization, data input/output, resource usages, etc., which also play a role in the definition of compliance requirements [20].

D. Problem Definition

The problem definition relies on the following two observations from industry practices: (i) the extraction of Compliance Requirements (CRs) in the form of textual snippets from a regulation, e.g., a law paragraph. It is up to the judgment of the regulation expert to decide upon the degree of granularity and size of CRs. (ii) These CRs can be refined together by regulation experts and business analysts until they reach a state where they explicitly refer to actions and elements defined in the enterprise model.

In our opinion, the paramount success criterion of a novel compliance management language is its capability to check the modeled CRs on business processes, detect violations, and proceed by handling these violations. We divide the RCM problem (see definition I.1) into two sub-problems.

Definition II.1 (RCM sub-problems)**.** ***1. Compliance Modeling:*** *The problem of (accurate) representation of CRs.*
2. Compliance Checking: *The problem of statically verifying whether or not a given enterprise model fulfills the CRs and/or the problem of enforcing CRs at runtime.* □

We propose to deal with the RCM problem by following a model-driven and policy based approach. This will allow us to build on existing approaches and languages in order to combine CR modeling, verification at design-time and enforcement at run-time.

III. Approach

This section addresses the most important aspects of our approach that are relevant for fully grasping the rationale behind the contribution of CoReL.

A. The Domain Divide in Compliance: Decision-Making Vs. Rule-Checking

In this paper we introduce CoReL, a compliance decision modeling language based on policies. In our view, there exists a divide between CRs and their implementation with the purpose of automatic checking. This divide comes mainly from the fact that in the process of compliance modeling, as well as compliance checking, several types of domain experts are involved. At a higher level of abstraction we find compliance experts including both regulation experts and business analysts. They have a good understanding of the regulation and its impact on the enterprise. For instance, compliance experts know what risks of non-compliance exist in the enterprise business models and work together with business analysts in order to come up with internal controls in order to tackle the latter [16]. On the other side, rule experts know about the different alternatives to implement a CR in a logical manner to make it amenable to automated checking. For example, a rule expert will know how to use OCL [7], Drools[9], or even a more complex temporal logic language (e.g., LTL[24]).

This divide can be expressed in terms of domain specific languages [25], as proposed in [26]. In this work we tackle this divide using CoReL by proposing a policy-based paradigm for modeling compliance decision making. Our approach introduces policies as an intermediate layer between CRs and the logic that implements them. We define a policy as follows:

Definition III.1 (Policy)**.** *A policy is a formal unit of decision making that declaratively models guidance on the behavior of a system. A policy uses rules to represent the logic behind the decision making.* □

B. A DSML for Compliance Decision Modeling

In this paper, as shown in the top part of Figure 1, we focus on proposing a solution to the compliance modeling problem, i.e., a Domain Specific Modeling Language (DSML) for compliance decision modeling called CoReL. In Model Driven Engineering (MDE), DSMLs are formally defined languages that focus on a specific domain of discourse. Their structure is defined as a metamodel together with consistency rules. The two together are referred to as the abstract syntax model of the DSML. Several concrete syntaxes (i.e., notations presented to the user) can be defined for the same DSML. Domain Specific Models (DSMs) are models specified in a DSML that satisfy the metamodel constraints. DSMLs are tailored to an intended use (i.e., for us, transformation and verification) and are much smaller than bigger scope languages (e.g., modeling languages such as UML, programming languages such as Java). CoReL is a DSML for *compliance decision-making*. In our approach, sketched in Figure 1, the input for the CoReL policy modeler are CRs, in the form of text snippets extracted from the tackled regulatory document (e.g., SOx). The CoReL modeler then creates a (set of) policy model(s) written in CoReL. This models specifies metadata about the policy, the context in which it applies, the control(s) it contains,

[9] jboss.org/drools

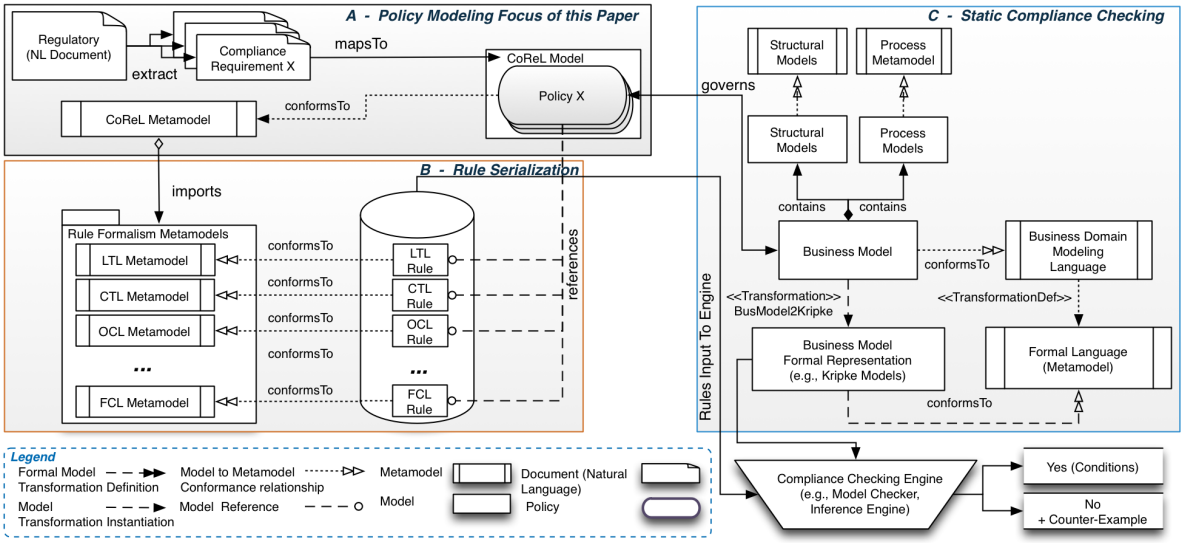


Figure 1. CoReL Approach - Model-Driven Policy-Based Compliance Modeling and Checking

the violation(s) it allows to deal with, and the appropriate reparation of the violation(s). We will get back to these elements in Section IV where we introduce CoReL in more detail. We also explain how rules are selected and embedded to implement a policy.

C. Rules in CoReL

CoReL has an agnostic view on rule languages. As is confirmed by previous research, several rule languages are of interest, due to different expressiveness and complexity properties [27]. From the point of view of CoReL, rule languages can be plugged in, and their interpreters used in the compliance checking process. This is out of the scope of this paper however, and we simply show in Section IV how rules can be embedded in CoReL policies. In CoReL, rules (i.e. rule statements) are extracted from policies by the means of a model transformation. We call this a policy *serialization*. To this purpose, every rule formalism plugged-into CoReL has a formal metamodel, and a model transformation formally describes how to *serialize* CoReL policies. This is shown in the left part of Figure 1.

Decoupling the domains of policies and rules has some advantages: (i) Policies are practical to group and manage rules. (ii) Decoupling policies from rules makes it easier for the business users to understand the way a regulation is modeled, and to cope with huge numbers of rules, as they can trace every rule to a policy. (iii) Rules are actionable pieces of logic written in different formalisms that allow automated reasoning. (iv) Rules can be reused and combined with other rules to implement new policies. We make use of this feature for multi-formalism policy verification.

D. Compliance Checking: Static and Dynamic

The eventual goal in the CoReL approach is automated compliance checking (C^2). Classically in research, three main scenarios for compliance checking have been considered: design-time (static), run-time (dynamic), analysis-time (post-execution) [20].The semantics given to CoReL currently allows two types of compliance checking: static and dynamic compliance checking.

1) Static Compliance Checking - C^2_S: At least two alternatives exist for C^2_S: verification by deductive reasoning or model checking. Model checking allows to formally verify rules written in temporal logic. Many of these logics, such as LTL, CTL [24], have been thoroughly studied in research. Several works in recent years have provided additional layers using transformations and graphical notation on top of model checking software in order to: (i) simplify its use, (ii) hide its complexity, and (iii) put the existing tooling and optimizations to profit [28], [18]. As shown in the right part of Figure 1 (block C), we use formal model transformations from the business model into a formal structure[10]. Both the formal structure and the serialization of CoReL policies in a temporal logic language are fed into a model checker, which outputs whether the temporal rule holds or no. In case it does not hold, a counter-example of a process model instance is provided.

2) Dynamic Compliance Checking - C^2_D: Many violations can only be uncovered at run-time, since much data, which depends on the current instance of the process, is only known during execution. A trivial case is that the person in

[10]Labelled Transition Systems, Finite State Machines are also other formalisms used.

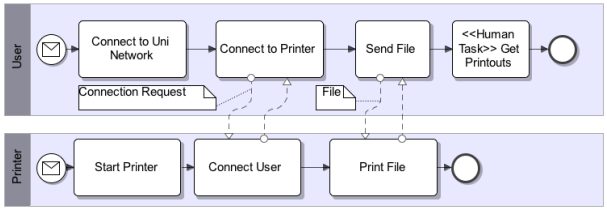


Figure 2. The printer access control example [In BPMN]

charge of executing a process activity may be only known at run-time, and may change from a process instance to the other. To tackle this, CoReL has formal operational semantics. These are outlined informally in Section IV, where we show how the *interpretation* of a CoReL policy is done. This interpretation uses the output of C_S^2, which are conditional statements allowing to evaluate CoReL policy rules to True or False at run-time. Such conditional statements can result from CoReL rules which cannot be evaluated by the model checkers, as is explained in Section IV. We left C_D^2 out of Figure 1 for the sake of simplification.

IV. The Compliance Representation Language (CoReL)

In the following, we introduce the central concepts of CoReL, beginning with the core concept of *Policy*. We also outline the informal semantics of a CoReL policy.

We will illustrate the use of CoReL on the example of a BPMN [29] process model given in Figure 2, defining the usage of a university printer by university users. The first pool is called "User" and shows a sequence of BPMN tasks that the user needs to execute in order to achieve his goal of printing a given file. The second pool is called "Printer" and shows the tasks that the printer will execute in order to fulfill the user's goal of printing a given file.
The file to be printed is transmitted between the two actors (represented by BPMN pools) of the interaction using BPMN messages[11]. The task "ConnectToPrinter" sends a message to the printer with a connection request and the printer replies with a message acknowledging the connection. In the following step, the task "SendFile" sends the file to be printed to the printer, and the printer replies with a confirmation that the file was correctly printed. The file is represented as a BPMN artifact.

A. CoReL Syntax - Policy Modeling Constructs

The abstract syntax model of CoReL is given in Figure 3. In order to ease the understanding we group the main language elements in three "blocks" shown in Figure 3: the *ASE Block* represents concepts related to the elements of the enterprise model for which the policy is defined, the *Decision Block* includes concepts that model how the compliance check is conducted, the *Reparation Block* concepts model how the compliance decision acts on the enterprise model.

First, we define the Policy concept, the core abstract concept of CoReL. Then, each of the graphical symbols used in the concrete syntax of CoReL is described, in the order given by the blocks drawn on Figure 3.

Definition IV.1 (CoReL Policy)**.** *A CoReL policy is the formal modeling of a CR as a decision to be made about whether to either* {allow, force, prohibit} *the execution of an Action by a Subject on an Entity.*

Policy

A *Policy* has a unique ID and a deontic modality (Permission, Interdiction, Obligation). Metadata objects are associated with the policy and allow to describe information about the policy such as the creator identity, the date of creation, date of first instantiation, risks it mitigates and internal controls it implements, etc. A one-to-one relationship links a CR to a *Policy*. A *Regulation* represents the origin of the CR modeled by the *Policy*.

1) ASE Block: We call $(Action, Subject, Entity)$ an *ASE-Triple* [30].

Subject
Action
Entity

The definition of an ASE-Triple maps *Action* to an enterprise model action, which in our approach applied to the business process domain, is a business process task/activity, and maps both *Subject* and *Entity* to any element of the enterprise model capable of executing the Action, resp. on which the Action is executed. Subject and/or Entity can possibly be defined as empty (nil). Definitions similar to our *ASE-Triple* concept exist in research around policy languages[11] but not in literature around enterprise regulatory compliance management (i.e., compliance defined for enterprise models).

2) Decision Block: The concepts of Context and Control (see below) build upon the concept of *Rule*, which does not have a dedicated symbol, as a Rule is embedded in a Context or a Control. Concretely, a Context and Control can be evaluated only if both the ASE-Triple and the Rule statement that refers to Action, Subject and Entity are provided. Otherwise, the policy cannot be interpreted and is not enforced. A Rule in CoReL can be written in various formalisms (LTL, CTL, FCL, OCL, etc.) and is denoted by: $R^{<Name>}_{<Formalism>}$, where $< Name >$ is the name identifying the Rule and $< Formalism > \in$ {'LTL', 'CTL', 'FCL', 'OCL', ...} specifies the formalism used to express the Rule. Given an ASE-Triple $(Action, Subject, Entity) \mapsto (SendFile, User, File)$, the following Rule is an FCL formular expressing that in order to execute the Action, there is an obligation on the Subject to execute the process task ConnectToUniPrinter: $R^3_{FCL} = Action \vdash O_{Subject} ConnectToUniPrinter$.

[11] BPMN messages are represented as dashed arrows with triangular empty heads that cross pools.

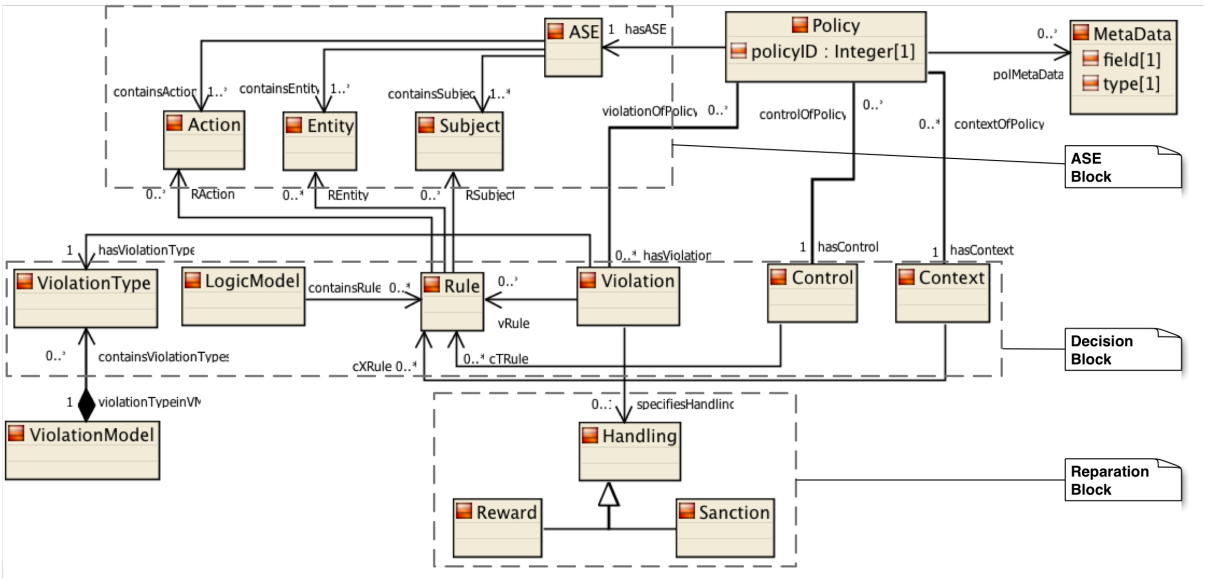
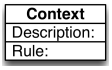


Figure 3. CoReL -CORE Metamodel [in Kermeta]

Rules used in the examples for context and control are given directly in a logical formalism. However, Rules are destined to be available as templates in a repository called *Rule Library* (cf. Figure 3). These templates are in the form of statements available in various formalisms supported by CoReL. A textual description always accompanies the definition of the rules in order to ease the task of business users. A current limitation of the approach is that a mechanism to find and select appropriate Rules for use in a CoReL Policy is required to make the modeling task easier.

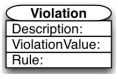


The *Context* of a policy formally models the state of the system in which the policy is applicable. Contexts are modeled as a conjunction of Rules. For instance, imagine that we want to express a CR on printer users who are students. This CR is only applicable on the weekends. For the ASE-Triple defined as $(Action, Subject, Entity) \mapsto (SendFile, User, File)$, we model a Policy P, with context X^P of P defined as:

Example IV.2 (Context)**.** $R^1_{Prop} = Weekend(Date) \wedge Today(Date) \wedge Student(Subject)$
$X^P = R^1_{Prop}$

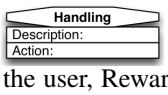
The *Control* part is a conjunction of rules and specifies the constraints that must hold on the enterprise model. For example, a CR says that a printer user can only print 2000 pages if (s)he is a faculty member. Moreover, it must be ensured that the printer only prints a file if the user requesting the print is connected to the university network. We define for the ASE-Triple defined as $(Action, Subject, Entity) \mapsto (SendFile, User, File)$ a Policy P, with control (denoted as T^P):

Example IV.3 (Control)**.** $R^1_{Prop} = Faculty(Subject) \rightarrow PagesLEQ2000(Subject)$
$R^2_{CTL} =$
$\boldsymbol{EG}(\boldsymbol{A}(ConnectToUniNetwork(Subject)\boldsymbol{U}Action(Entity)))$
$T^P = R^1_{Prop} \wedge R^2_{CTL}$



Several *Violations* can be modeled for a Policy, each of which maps an evaluation of the Rules in the Policy Control part to a violation value of a violation type. The available violation types are pre-defined in a specific model, called violation model, which is application specific and can be defined by CoReL users. For example, given a Policy P, let the Control of P be the conjunction of two Rules. P's Violations carry a violation value belonging to the enumeration: $\{Red, Yellow, Green\}$. The user models a Violation of value Red if the two Rules in the Control part of the Policy both return False, a Violation of value $Yellow$ if one of the two Rules returns True and the other False, and a Violation of value $Green$ if both Rules return True.

3) Reparation Block: For each Violation, CoReL allows to specify a step called violation reparation which serves the need to trigger additional actions in order to deal with or prevent such violation from occurring, or to encourage it to happen again.



Every Violation has a (possibly empty) set of *Handling* (*Rewards* or *Sanctions*). To the user, Rewards are used to model how to react positively, and Sanctions, on the opposite, are used to model negative reactions based on the violation value. Note that in CoReL, Rewards and Sanctions are modeled in the same way, the

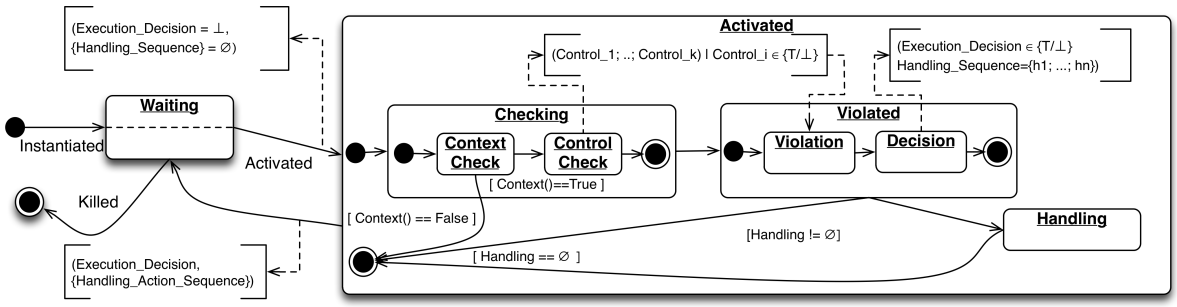


Figure 4. Policy Behavioral Semantics - [UML State Chart]

only concrete syntax difference is the color of the symbol lines: red for Sanctions and blue for Rewards.

Rewards and Sanctions always trigger some specific informative action, such as the execution of a function that calculates penalty points, or an action that sends notification emails. For example, the user can assign a Reward to a Violation with value $Green$ that increases some rating given to the user of the printer, or assign a Sanction to the printer user that decreases its rating if the Policy decides that he caused a Red Violation. The actions used in the definition of Handling are application-specific, provided by the business model and can be selected by the CoReL modeler.

B. CoReL Semantics - Policy Interpretation

In Figure 4, the lifecycle of a Policy is modeled. We refer to the external software environment responsible for calling the Policy and processing the result of the Policy decision as the *System*. Such a system can be for example the business process execution engine which executes business process models enriched with CoReL Policies. The system decides whether to execute a certain action or not, and asks the Policy to take a decision about this.

Initially, when a Policy is instantiated by the system, it is in the *Waiting* state, and it awaits activation. When the Policy must be checked, i.e., the ASE-Triple is met, it is *Activated*. Inside the super state *Activated*, the Policy moves into the *Checking* state, where, in case its Context is evaluated to True, the Policy will check its Control part. After checking the Control part, the Policy moves into the *Violated* state, where a violation value is computed in state *Violation* and mapped to an *execution decision* in state *Decision*. In addition, the corresponding violation handling actions are triggered in state *Handling*.

The execution decision of a Policy is of the form True/False, where True means that the system should resume process execution and execute the action, and False means halting process execution and not executing the action. By default, the execution decision is set to False when a policy is activated. Note that the mapping from violation value to execution decision is application-specific and must be defined by the user. For example, the user may allow execution in case of $Green$ and $Yellow$ violation values, and forbid so in case of a Red violation value.

Compliance in CoReL In CoReL, the decision taken about whether the process model is compliant or not is represented by the boolean execution decision. Moreover, the interpretation of a CoReL policy also triggers a sequence of handling actions whose execution is enforced by the process execution engine. In the case where two policies expressed on the same ASE-Triple lead to a conflicting execution decision, CoReL's current approach is defensive, and gives precedence to the negative compliance decision, i.e., the execution decision is False.

V. Modeling A Simple Business Process Printer Management Regulation

In the running example shown in Figure 2, we have two concurring business process pools. One models the process followed by a printer, and the other represents the behavior of a printer user. The system administrator, after reading the regulation, defines a list of CRs based on his knowledge and experience.

Table I
Compliance Requirement Examples

CR	Definition
CR1	External students cannot print on university printers. regular and exchange students can.
CR2	Students are prohibited from printing on the week-end. Faculty members can.
CR3	Students are limited to 400 pages a month, with a bonus of 50 pages if it is their first month at university. Every student who prints less that 30% of his allocated print pages over a whole semester can receive a 5 euro waiver ticket to use the university's swimming pool.
CR4	faculty members can print up to 2000 pages a month. Faculty members are allowed a violation of the max number of pages if it amounts to less than 10%, in which case they are warned per email. In case they exceed the number of allocated pages by more than 10% pages, their access to printers is stopped.

Table I gives the requirements defined by the system administrator that we represent using the CoReL model in

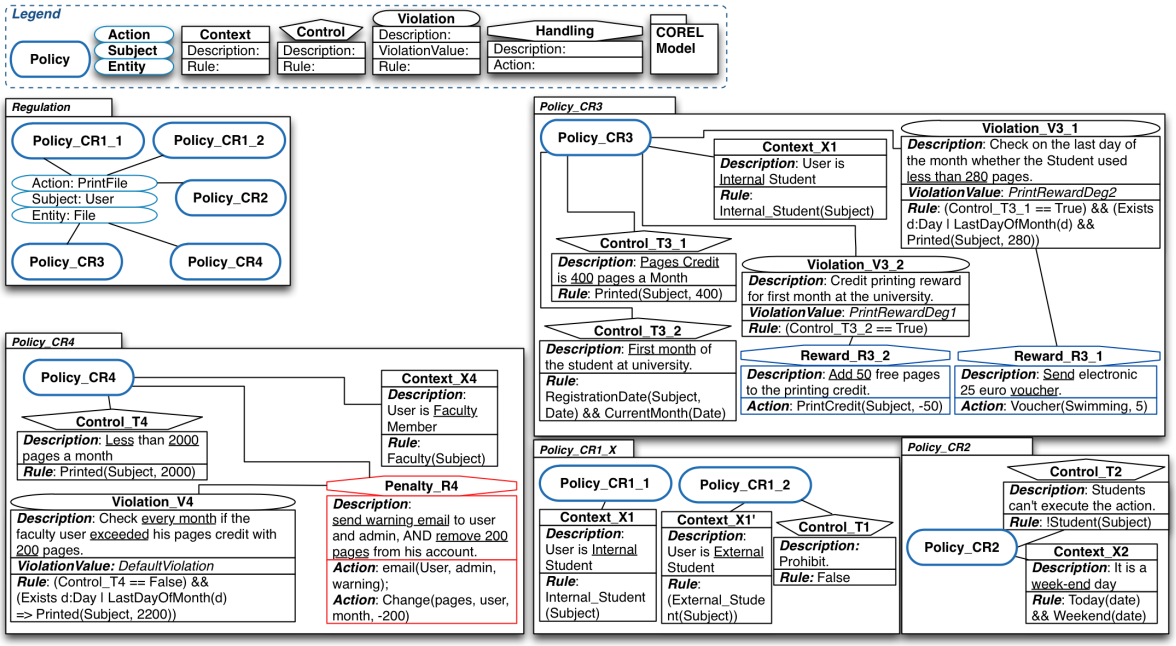


Figure 5. CoReL - Printer Case Study - CoReL Models for the Regulation (i.e., Set of CRs)

Figure 5. The various CoReL elements can be easily reused (e.g., an *ASE-Triple* or a Violation is reused by several Policies). The reader will notice that the Context_X1 is shared by both Policy_CR1_1 and Policy_CR_3. The information needed to evaluate the Policy parts comes from the business model (e.g., business process model). The content of the BPMN message is defined as an XML file, and specifies the file content to be printed and information about the user who sends it. For lack of space we do not show the external models here.

The main model is the regulation model, representing an ASE-Triple and the Policies governing it. Note that Policy_CR1 is modeled as the conjunction of two sub-policies: Policy_CR1_1 and Policy_CR1_2. The semantics of this operation is similar to the boolean AND operation. Policy operations are defined in an algebra of Policies, which is not discussed in this paper.

In case a Policy does not specify a Violation, the default binary Violations are assumed: one corresponds to the case where the Control evaluates to True, i.e., all Rules in the Control hold, and the other corresponds to the case where the Control evaluates to False, i.e., at least one Rule in the Control is violated. The execution decision associated with the first default Violation is True and the second False. Also, in case a Policy does not specify a Control, the default Control, namely True is assumed.

Currently, CoReL does not provide a graphical way of modeling the logic behind a Rule. For this reason, we embed every Rule in the Context/Control as a string written in a given formalism. In future work, we plan to build on work on the BPSL graphical notation by Liu, Xu et al. [18], [31] and adapt it to the needs of CoReL.

VI. Discussion

In the following paragraphs, we discuss how CoReL addresses the shortcomings of enterprise compliance management approaches identified in Section II-C.

1) Business-User Orientation: CoRel is a DSML for compliance experts. It allows them to focus their attention on how to model compliance decisions based on the intuitive concept of policies. The use of the DSML is facilitated by its graphical concrete syntax.

2) Modularity: CoReL's constructs are meant to be used as building blocks to create policies more easily and quickly. For instance, a Context element in a CoReL policy can be reused to help build another CoReL policy. In fact, the intended use of CoReL is such that an enterprise would ideally have a library of CoReL modeling elements (i.e., modelled Contexts, Controls, Violations, etc.) that might thus be selected and used by CoReL experts to create new policies or update policies.

3) Flexibility: In CoReL compliance requirements are decoupled from the business model. Hence, when a regulation changes only the corresponding compliance requirements and the supporting policies need to be modified; no modifications to the business model are necessary.

4) Multiple Business Domains: CoReL does not contain any concept relating to a specific business domain to which

CoReL policies might apply. There exists a variety of languages for such business domains: enterprise modeling (e.g., ARIS [32]) or business process modeling languages (e.g., BPMN [29], EPC [33]). The language constructs that allow linking CoReL policies to business domain models are the ASE triples (Action, Subject, Entity), and the rules implementing the constraints expressed by policies.

5) Multiple Regulation Types: Enterprise compliance covers multiple types of regulations. These range from security, quality, financial, medical, human resources to contracts or governance regulations. CoReL is expected to be used in the same fashion to represent compliance requirements extracted from different types of regulations.

6) Multiple Logical Formalisms: The same argument goes for expressing compliance requirement constraints. In CoReL, Rules used in Contexts and Controls express the logical structural or behavioral constraints. For instance, it shall be allowed to combine rules written in OCL with other rules written in LTL or CTL. The evaluation of the rules to True or False is outsourced to the rule engines capable of interpreting (e.g., OCL) or model checking (e.g. CTL) these rules.

7) Traceability: As it is possible to directly link a CoReL policy to the compliance requirement from which it originates and thus to the regulation it partially implements, traceability is ensured. This can be modeled as a trivial query to the policy model written in CoReL (e.g., using OCL queries).

8) Holisticness: CoReL does not only consider process models or organizational models: CoReL rules can include any business model element as a variable through the use of the Action, Subject and Entity keywords. CoReL abstracts from business models by including both elements and actions. Actions can be bound to processes and process activities. Elements can be bound to structural aspects such as data, resources, and roles.

VII. Final Thoughts

A. Future Work

Our first task in defining the CoReL DSML consisted of fully defining the abstract syntax model of the language, including consistency properties (well-formedness rules). We implemented it as a Kermeta[12] model. Kermeta is a tool-supported (eclipse-based) language for formal metamodelling allowing to define model transformations. The Kermeta language is based on EMOF, so our metamodels are available as EMF[34] models. We are working on an implementation of the concrete syntax of CoReL using GMF[13], in order to generate an ECORE-based editor for CoReL. Our validation step will thereafter address three issues: (i) the expressiveness of the language based on chosen regulation examples; (ii) the suitability of CoReL to business users; this will require an empirical study which involves target users; (iii) and finally, the possibility of modeling formal transformations from CoReL models to selected target verification languages. We plan to showcase the validity of the approach for LTL, CTL[24], OCL[7] and FCL ([35]) languages. We are currently completing the formal semantics of CoReL for C_S^2 and C_D^2 which includes other features of the language not mentioned in the paper, e.g., a policy algebra allowing to create complex policy expressions.

[12]http://kermeta.org/

[13]http://www.eclipse.org/modeling/gmp/

B. Conclusion

The contribution of this paper is the CoReL language, which is designed for compliance modeling. CoReL's aim is to allow for modular modeling of compliance decision-making that seeks to achieve flexible compliance, in opposition to strict binary compliance. We motivated the use of policies as core concepts for modeling compliance requirements. Policies form an abstraction layer on top of rules, allowing to cover several verification formalisms. We place the compliance management problem in a context where it is targeted at business users, needs to deal with enterprise models, and be modeled in and enforced on business processes. CoReL's approach is model driven, which allows it to fulfill several of the requirements posed because of the particular context in which we place compliance management. Also, MDE enables the multi-formalism verification of CoReL models using model transformations. We expect these properties of CoReL to prove a real advantage in an environment where business users need to deal with regulations.

References

[1] J. Jürjens, "Umlsec: Extending uml for secure systems development," in *UML2002 - The Unified Modeling Language*, springer berlin / heidelberg ed., 2002, vol. 2460.

[2] M. El Kharbili and E. Pulvermueller, "Service contract compliance management," in *Proceedings of the 3rd Workshop on Emerging Web Services Technology*, Dublin, Ireland, November 2008, pp. 87–96.

[3] S. Sadiq and G. Governatori, "Managing regulatory compliance in business processes," in *Handbook on Business Process Management 2*, J. Brocke and M. Rosemann, Eds., 2010, pp. Springer Berlin Heidelberg–175.

[4] S. A. Norris, S. Sadiq, and M. Indulska, "Emerging challenges in information systems research for regulatory compliance management," in *Advanced Information Systems Engineering*, B. Pernici, Ed., 2010, vol. 6051, p. Springer Berlin / Heidelberg.

[5] M. El Kharbili, A. K. Alves de Medeiros, S. Stein, and W. M. P. van der Aalst, "Business process compliance checking: Current state and future challenges," in *Proceedings of MobIS*, ser. LNI, vol. 141, 2008, pp. 107–113.

[6] S. Turki and M. Bjekovic-Obradovic, "Compliance in e-government service engineering: State-of-the-art," in *Exploring Services Science*, W. A. et al., Ed., vol. 53, 2010.

[7] OMG, "Object constraint language," OMG Available Specification, Version 2.0, May 2006. [Online]. Available: http://www.omg.org/docs/formal/06-05-01

[8] E. C. Jr., O. Grumberg, and D. Peled, *Model Checking*. MIT Press, Cambridge, 1999.

[9] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," *Lecture Notes in Computer Science*, vol. 1995, pp. 18–??, 2001.

[10] A. Uszok and J. e. a. Bradshaw, "Kaos policy management for semantic web services," *Intelligent Systems, IEEE*, vol. 19, Issue: 4, no. ISSN: 1541-1672, pp. 32– 41, 2004, jul-Aug 2004.

[11] L. Kagal., "A policy-based approach to governing autonomous behavior in distributed environments." PhD Thesis, Faculty of the Graduate School of the University of Maryland, 2004.

[12] B. P. A. Bonatti: REWERSE project I2-D1 deliverable Piero A. and S. e. a. Nahid, "Rule-based policy specification: State of the art and future work," 4 September 2004.

[13] C. Project, "State-of-the-art in the field of compliance languages," Tilburg University, http://www.compas-ict.eu/results.php, Deliverable D2.1, 2008.

[14] G. Governatori, Z. Milosevic, and S. Sadiq, "Compliance checking between business processes and business contracts,," in *Proceedings of the 10th IEEE International EDOC*, 2006, pp. 221–232.

[15] J. Hoffmann, I. Weber, and G. Governatori, "On compliance checking for clausal constraints in annotated process models," *Information Systems Frontiers*, pp. 1–23, 2009.

[16] K. Namiri, "Model-driven management of internal controls for business process compliance," Ph.D. dissertation, University of Karlsruhe, 2008.

[17] L. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam, "On enabling integrated process compliance with semantic constraints in process management systems," *Information Systems Frontiers*, pp. 1–25, 2009.

[18] Y. Liu, S. Mueller, and K. Xu, "A static compliance-checking framework for business process models," *IBM Syst. J.*, vol. 46, no. 2, pp. 335–361, 2007.

[19] W. M. van der Aalst, "Matching observed behavior and modeled behavior: An approach based on petri nets and integer programming."

[20] M. El Kharbili, S. Stein, I. Markovic, and E. Pulvermueller, "Towards a framework for semantic business process compliance management," in *Proceedings of the GRCIS workshop.*, S. S. et al., Ed., vol. 339, 2008, pp. 1–15. [Online]. Available: http://CEUR-WS.org/Vol-339/

[21] E. Bertino, P. A. Bonatti, and E. Ferrari, "Trbac: A temporal role-based access control model," *ACM Trans. Inf. Syst. Secur.*, vol. 4, pp. 191–233, August 2001.

[22] Goedertier, Mues, and Vanthienen, *Specifying Process-Aware Access Control Rules in SBVR*, 2007, pp. 39–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75975-1_4

[23] G. Boella, G. Governatori, A. Rotolo, and L. van der Torre, "A formal study on legal compliance and interpretation," in *13 International Workshop on Non-Monotonic Reasoning (NMR 2010)*. CEUR Workshop Proceedings, 2010.

[24] B. Christel and J.-P. Katoen, *Principles of Model Checking*. MIT Press, Cambridge, 2008.

[25] A. Kleppe, *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*. Addison-Wesley Professional, 2008, no. 978-0321553454.

[26] E. Oberortner, U. Zdun, and S. Dustdar, "Tailoring a model-driven quality-of-service dsl for various stakeholders," in *ICSE Workshop on Modeling in Software Engineering*. Vancouver, Canada: IEEE, May 2009.

[27] A. Elgammal, O. Turetken, W. van den Heuvel, and M. Papazoglou, "On the formal specification of regulatory compliance: A comparative analysis," in *Proceedings ICSOC10 workshops*, M. et al., Ed. Springer, 2010.

[28] E. Pulvermueller, S. Feja, and A. Speck, "Developer-friendly Verification of Process-based Systems." *Journal on Knowledge-Based Systems (KNOSYS)*, vol. 23, no. 7, pp. 667 – 676, October 2010.

[29] OMG, "Business process modeling notation (bpmn) specification," Object Management Group (OMG), Tech. Rep., February 2011, http://www.omg.org/spec/BPMN/2.0/PDF.

[30] M. El Kharbili and E. Pulvermüller, *Semantic Technologies for Business and Information Systems Engineering: Concepts and Applications*. IGI Global, 2011, ch. Semantic Policies for Modeling Regulatory Process Compliance.

[31] K. Xu, Y. Liu, and C. Wu, "Bpsl modeler - visual notation language for intuitive business property reasoning," *Electronic Notes in Theoretical Computer Science*, vol. 211, pp. 211 – 220, 2008.

[32] A.-W. W. Scheer, *Aris-Business Process Frameworks*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998.

[33] W. M. P. van der Aalst, "Formalization and verification of event-driven process chains," *Information & Software Technology*, vol. 41, no. 10, pp. 639–650, 1999.

[34] S. Dave, B. Frank, P. Marcelo, and M. Ed, *EMF Eclipse Modeling Framework*. The Eclipse Series, Addison Wesley, 2009.

[35] G. Governatori and M. Z, "Dealing with contract violations: formalism and domain specific language." in *Proceedings of the Conference on Enterprise Computing EDOC 2005*. IEEE Press., 2005, pp. 46–57.