Regorous: A Business Process Compliance Checker

Guido Governatori, Sidney Shek
NICTA, Australia*
{guido.governatori,sidney.shek}@nicta.com.au

ABSTRACT

We report on the development of Regorous, a business process compliance checker, based on the compliance-by-design methodology proposed by Governatori and Sadiq [8]. For a screencast see http://www.youtube.com/watch?v=gFmDQJNai_4r

1. INTRODUCTION

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 [3] proposed that *business process compliance* is a relationship between the formal representation of a process model and the formal representation of a relevant regulation. Compliance strategies can be classified as *detective*, *corrective* and *preventative* [11].

Detective measures are intended to identify an "after-the-fact" un-compliant situation. There are two main approaches: (a) *retrospective reporting* through manual audits by consultants or through IT forensics and Business Intelligence tools; (b) *automated detections* generating audit reports against hard-coded checks performed on the requisite system. Unlike the first approach, automated detection reduces the assessment time and consequently also the time of un-compliance remediation/mitigation.

Corrective measures are intended to limit the extent of any consequence caused by an un-compliant situation. That situation can arise in front of the introduction of a new norm impacting upon the business, to the organisation coming under surveillance and scrutiny by a control authority or to an enforceable undertaking.

The two approaches above suffer from lack of *sustainability*, caused by the extreme interest of companies to continuously improve the quality of service, and for changing legislatures and compliance requirements. Even with automated detection means, the hard coded check repositories can quickly grow to a very large scale making it extremely difficult to evolve and maintain. To obviate these problem [10] proposes a *preventative focus* based on the idea of *compliance-by-design*, namelly: to supplement business process models with extra information to ensure that processes are compliant with a relevant normative framework before their deployment.

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

ICAIL'13 Jun 10-14 2013, Rome, Italy
ACM 978-1-4503-2080-1/13/06.

2. REGOROUS ARCHITECTURE

Regorous is a business process compliance checker based on the methodology proposed in [8]. 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. For the regulation we use PCL [2, 6, 7]. PCL is a simple, efficient, flexible rule based logic combining defeasible logic (for the efficient and natural treatment of exceptions which are common in normative reasoning) [1] and a deontic logic of violations [4]. In FCL norm are represented by rules with the form

$$a_1, \ldots, a_n \Rightarrow c,$$

where $a_1, \ldots, a_n$ are the conditions of applicability of the norm/rule and c is the *normative effect* of the norm/rule. FCL distinguishes two normative effects: the first is that of introducing a definition for a new term. Thus for example the rule

$$customer(x), spending(x) > 1000 \Rightarrow premium\_customer(x)$$

specifies that typically a premium customer is a customer who has spent over 1000 dollars. The second normative effect is that of trigger obligations and other deontic notions. The deontic notions covered by FCL are obligations[1], permissions, and reparation chains. For obligations FCL supports both maintenance obligations and achievement obligations, and for achievement obligations both pre-emptive and non-pre-emptive obligations (see [6] for full details). A reparation chair is an expression $O_1c_1 \otimes O_2c \otimes \cdots \otimes O_nc_n$, where each O_i is an obligation, and each c_i is the content of the obligation. The meaning of a reparation chain is that we have c_1 is obligatory, but if c_1 is violated, i.e., we have $\neg c_1$, then the violation is compensated by c_2 (which is then obligatory). But if even O_2c_2 is violated, then this violation is compensated by c_3 which, after the violation of c_2, becomes obligatory, and so on.

It is worth noticing that PCL allows deontic expressions (but not

[1]Note that obligations allow us to capture prohibitions; a prohibition is an obligation plus negation, for example the prohibition to smoke can be understood as the obligation not to smoke.

reparation chains) in the body of rules, thus we can have rules like:

$$cafe, [P]sell\_alcohol \Rightarrow [OM]show\_license \otimes [OAPNP]pay\_fine$$

The rule above means that if a cafe has a license to sell alcohol (i.e, it is permitted to sell it, [P]*sell_alcohol*), then it has a maintenance obligation to expose the license ([OM]*show_license*), if it does not then it has to pay the fine ([OAPNP]*pay_fine*). The obligation to pay the fine is non-pre-emptive (this means it cannot be paid before the violation). PCL is agnostic about the nature of the literals it uses. They can represent tasks (activities executed in a process) or propositions representing state variables. For full descriptions of PCL and its feature see [2, 6, 7].

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 artifacts related to the process, and the resources linked to the process. Accordingly, process models must be enriched with such information. [12] proposes to enrich process models with semantic annotations. Each task in a process model can have attached to it a set of semantic annotations. In our approach the semantic annotations are literals in the language of PCL, representing the effects of the tasks.

Given an annotated process and the formalisation of the relevant regulation, the algorithm proposed in [5, 6] determines whether the annotated process model is compliant. 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. This is done by a call to an FCL 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.

A process is compliant iff all traces are compliant (all obligations have been fulfilled or if violated they have been compensated). A process is weakly compliant if at least one trace is compliant.

3. IMPLEMENTATION AND EVALUATION

Regorous is implemented on top of Eclipse. For the representation of process models, it uses the Eclipse Activiti BPMN 2.0 plugin, extended with features to allow users to add semantic annotations to the tasks in the process model. Regorous is process model agnostic, this means that while the current implementation is based on BPMN all Regorous needs is to have a description of the process and the annotations for each task. A module of Regorous take the description of the process and generates the execution traces corresponding to the process. After the traces are generated, it implements the algorithm outlined in the previous section, where it uses the SPINdle rule engine [9] for the evaluation of the PCL rules. In case a process is not compliant (or if it is only weakly compliant) Regorous reports the traces, tasks, rules and obligations involved in the non compliance issues.

Regorous was tested against the novel Australian Telecommunication Consumers Protection Code. Specifically, the section on complaints was manually mapped to PCL. The section of the code contains approximately 100 commas, in addition to approximately 120 terms given in the Definitions and Interpretation section of the code. The mapping resulted in 176 PCL rules, containing 223 PCL (atomic) propositions, and 7 instances of the superiority relation. Of the 176 rules 33 were used to capture definitions of terms used in the remaining rules. Mapping the section of the code required all features of PCL: all types of obligations apart punctual obligations were used, reparation chains, permissions, defeasibility to capture exceptions, and obligations and permissions in the body of rules.

The evaluation was carried over in cooperation with an industry partner operating in the sector of the code. 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. 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. We were able to identify non compliance issues in the processes and to rectify them. 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 MacBook Pro 2.2Ghz Intel Core i7 processor with 8GB of RAM (limited to 4GB in Eclipse).

References

[1] G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.

[2] G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.

[3] G. Governatori, Z. Milosevic, and S. Sadiq. Compliance checking between business processes and business contracts. In *EDOC 2006*, pp 221–232. IEEE Computing Society, 2006.

[4] G. Governatori and A. Rotolo. Logic of violations: A Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic* 4:193–215, 2006.

[5] G. Governatori and A. Rotolo. An algorithm for business process compliance. In *Jurix 2008*, pp. 186–191. IOS Press, 2008.

[6] G. Governatori and A. Rotolo. A conceptually rich model of business process compliance. In *APCCM 2010*, CRPIT 110, pp. 3–12. ACS, 2010.

[7] G. Governatori and A. Rotolo. Norm Compliance in Business Process Modeling. In *RuleML 2010*, LNCS 6403, pp. 194-209. Springer, 2010.

[8] G. Governatori and S. Sadiq. The journey to business process compliance. In J. Cardoso and W. van der Aalst (eds) *Handbook of Research on BPM*, pp. 426–454. IGI Global, 2009.

[9] H.-P. Lam and G. Governatori. The making of SPINdle. In *RuleML 2009*, LNCS 5858, pp. 315–322. Springer, 2009,

[10] R. Lu, S. Sadiq and G. Governatori. Compliance Aware Business Process Design. In *BPM Workshop 2007*, LNCS 4928, pp. 120-131. Springer, 2007.

[11] S. Sadiq and G. Governatori. Managing regulatory compliance in business processes. In J. van Brocke and M. Rosemann (eds) *Handbook of Business Process Management*, v.2, pp 157–173. Springer, 2010.

[12] S. Sadiq, G. Governatori, and K. Naimiri. Modelling of control objectives for business process compliance. In *BPM 2007*, LNCS 4714, pp 149–164. Springer, 2007.