A Short Introduction to the Regorous Compliance by Design Methodology

Guido Governatori
Data61, CSIRO, Australia
guido.governatori@data61.csiro.au

Abstract

We provide a short outline of the compliance by design methodology proposed by Sadiq and Governatori to ensure compliance of business processes and its implementation in Regorous.

1 Introduction

Many definitions of compliance have been proposed. However, regulatory compliance is generally understood as the set of activities in place in an organisation to ensure that the procedures, policies, processes and operations of the organisation comply with the normative frameworks governing the business and environment where the organisation operates. Sadiq, Governatori, and Naimiri [26] proposed a definition of regulatory compliance as a relation between two sets of specifications, more precisely, between the (formal) specifications of a set of regulations and the (formal) specifications of a system. The definition is further operationalised by focusing on formal models of business processes for the specifications of a system, and the relationship is that activities performed to execute a business process do not violate the (relevant) set of norms.

In the past decade a plethora of compliance frameworks have been proposed; we refer to [17, 25] for introductory material to general ideas about compliance, [7, 4, 2] for surveys and evaluations from the functionalities and business point of view, and to [20, 19] for classifications of the underlying approaches and their suitability to properly represent normative requirements. The *compliance-by-design* methodology and techniques advanced by [26], extended in [15, 17] and then implemented as Regorous [18, 10] emerged as a strong ICT based solution for handling the regulatory compliance of business processes [20, 19]; Regorous combines a conceptually sound logical representation of norms with mathematical models of business processes and, at the same time, it offers a practical solution to determine whether business processes comply with relevant normative frameworks.

In the next section we give a very brief overview of business processes and we argue that they can be used to represent a large class of systems. Then in Section 3 we discuss how to model norms in FCL, the rule language used in Regorous, then in Section 4 we outline the compliance architecture for Regorous.

2 Business Processes

A business process model is a self-contained, temporal and logical order in which a set of activities are expected to be 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 (we refer to [6] for an extensive presentation of business processes languages and modelling techniques, and [1] for the technical foundations). Besides the difference in notation, purposes, and expressive power, business process languages typically contain the following minimal set of elements: *tasks*, *connectors* (control flow gateways) and *events*. A task corresponds to a (complex) business activity, and connectors (e.g., sequence, and-join, and-split, (x)or-join, (x)or-split) define the relationships among tasks to be executed; for the events we restrict ourselves to the start and end event. The combination of tasks and connectors defines the possible ways in which a process can be executed. Where a possible execution, called *process trace* or simply *trace*, is a sequence of tasks and events respecting the order given by the connectors. We will use $\mathcal{T}_P$ to refer to the set of traces of a process P. In other terms, $\mathcal{T}_P$ represents all possible ways in which the process can be executed.

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 artifacts produced by the tasks (for example, the data resulting from executing a task or modified by the task) [21]. To capture this aspect [26] proposed to enrich process models with semantic annotations. Each task in a process model can have attached to it a set of semantic annotations. An annotation is just a set of formulas giving a (partial) description of the environment in which a process operates. Then, it is possible to associate to each task in a trace a set of formulas corresponding to the state of the environment after the task has been executed in the particular trace. Notice, that different traces can results in different states, even if the tasks in the traces are the same. In addition, even if the end states are the same, the intermediate states can be different. Accordingly, we extend the notion of trace. First of all, we introduce the function

$$State\colon \mathcal{T}_P \times \mathbb{N} \mapsto 2^{\mathcal{L}},$$

where $\mathbb{N}$ is the set of natural numbers and $\mathcal{L}$ is the set of formulas of the language used to model the annotations.

3 Modelling and Reasoning with Norms

As we have already discussed 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.

For the formal representation of the regulation we use FCL [9, 15]. FCL is a simple, efficient, flexible rule based logic. FCL has been obtained from the combination of defeasible logic (for the efficient and natural treatment of exceptions, which are a common feature in normative reasoning) [3] and a deontic logic of violations [16]. In FCL a norm is represented by a rule

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

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; the second is

The second normative effect is that of triggering obligations and other deontic notions. FCL supports all deontic notions (normative requirements) proposed and classified for compliance purposes in [8, 22], see Figure 1. In addition it has mechanisms

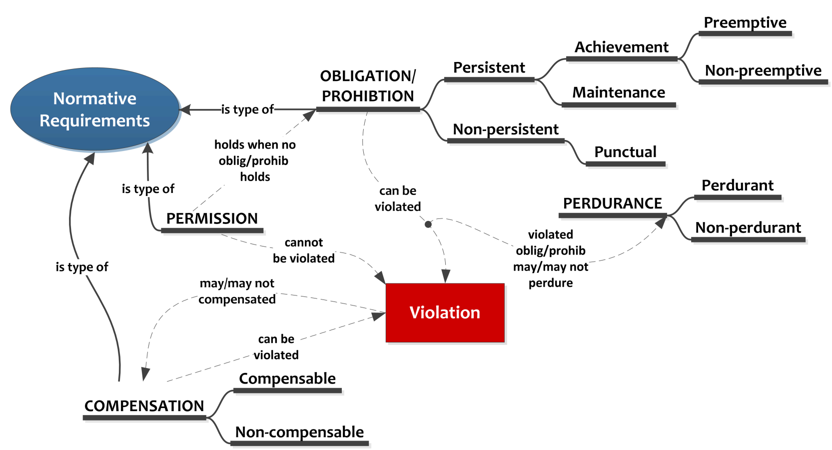

Figure 1: Normative requirements classification

to terminate and remove obligations (see [15] for full details). For obligations and permission we use the following notation:

- [P]p: p is permitted;
- [OM]p: there is a maintenance obligation for p;
- [OAPP]p: there is an achievement preemptive and perdurant obligation for p;
- [OAPNP]p: there is an achievement preemptive and non-perdurant obligation for p;
- [OANPP]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.

Compensations are implemented based on the notion of ‘reparation chain’ [16]. A reparation chair is an expression

$$\mathrm{O}_1 c_1 \otimes \mathrm{O}_2 c \otimes \cdots \otimes \mathrm{O}_n c_n,$$

where each O_i is an obligation, and each c_i is the content of the obligation (modelled by a literal). The meaning of a reparation chain is that we have that c_1 is obligatory, but if the obligation of 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 $\mathrm{O}_2 c_2$ is violated, then this violation is compensated by c_3 which, after the violation of c_2, becomes obligatory, and so on.

The reasoning behind FCL is the standard mechanism of defeasible logic [3] extend to handle the deontic notions. We refer the interested readers to [14] for a full description and all technical details.

4 Compliance

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 FCL[1], representing the effects of the tasks. Figure 2 depicts the architecture of the compliance methodology

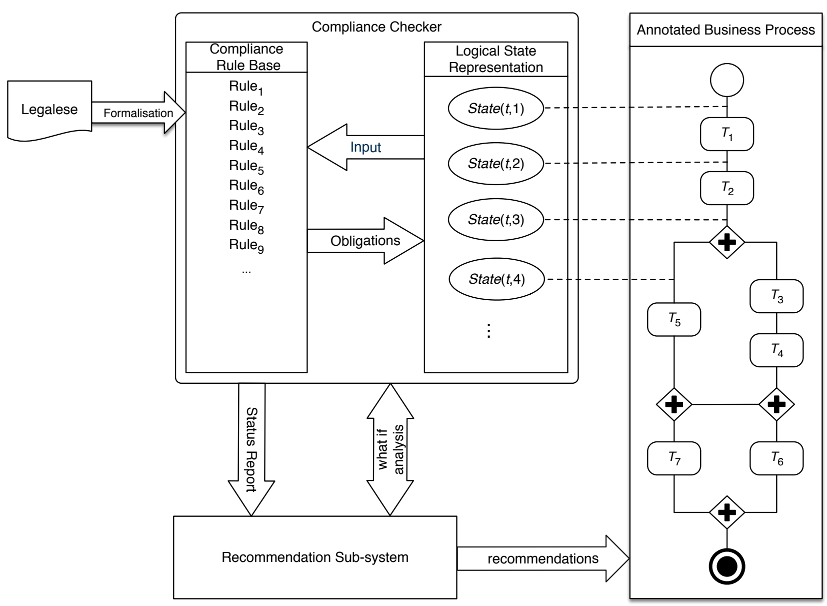

Figure 2: Compliance Checking Architecture

as implemented in Regorous. Given an annotated process and the formalisation of the relevant regulation, we can use the algorithm initially proposed in [15] to determine whether the annotated process model is compliant. The procedure runs as follows:

[1]FCL is agnostic about the nature of the literals it uses. They can represent tasks (activities executed in a process), or propositions representing state variables and the happening of events.

- 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 (fully) compliant if and only if all traces are compliant (all obligations have been fulfilled or if violated they have been compensated). A process is partially compliant if there is at least one trace that is compliant.

Regorous proved to be conceptually sound for the modelling of norms [10] against underlying semantics advanced in [8, 22] It does not suffer from the problem of wrong representation of norms affecting formalism based on possible world semantics [11] and temporal logic based compliance frameworks [13].

Notice that the Regorous's strategy to examine all traces (with the proviso that all loops are unfolded once) is optimal, in the sense that [5] proved that the problem of determining whether a process is weakly compliant is NP-complete and CoNP-complete for the case of full compliance.

It is worth noting that the core notion we require is that of a trace, which is can be simply understood as a sequence of relevant event. Thus, the idea presented in this section and used in Section 4 can be adopted for a large variety systems (all we need is a sequence of events, each with a set of semantic annotations attached to each event). Also, it can be used at different stages of the life-cycle of a business processes design-time, run-time and auditing. At design time, we have to simulate all possible execution of the process with abstract data, at run time, we use the data generated by an instance (case) of the process, and we can evaluate if up to the current task the process is compliant, determine what obligations are currently in force and predict if the instance is going to be compliant. Finally, it is possible to replay the entire log (split in cases) for auditing purposes, eventually retrieving the data from log of from the databases related to the processes [21, 23].

4.1 Implementation and Evaluation

As we have already alluded to our compliance-by-design methodology has been implemented in Regorous which has been 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 above, where it uses the SPINdle rule engine [24] for the evaluation of the FCL 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 successfully tested in a number of pilot project with industry partners in banking, insurance, telecommunications and building sectors. See [25, 18] for the results of the evaluation in the telecommunication sector for compliance against the Australian Customers Protection Code (C628-2012). As we noted above, in general, the problem of determine whether a business process is compliant is computationally intractable. However, Regorous adopts a series of strategies to reduce the number of computations [12]. Furthermore, experience with the pilot project seem to indicate the theoretical computational limits, do not really affect practical applications.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
2. Abdullah, N.S., Sadiq, S., Indulska, M.: Emerging challenges in Information Systems Research for Regulatory Compliance Management. In: CAiSE'10, LNCS, vol. 6051, pp. 251–265. Springer, Heidelberg (2010)
3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation Results for Defeasible Logic. ACM Transactions on Computational Logic 2(2), 255–287 (2001)
4. Becker, J., Delfmann, P., Eggert, M., Schwittay, S.: Generalizability and Applicability of Model-Based Business Process Compliance-Checking Approaches – A State-of-the-Art Analysis and Research Roadmap. BuR - Business Research Journal 5(2), 221–247 (2012)
5. Colombo Tosatto, S., Governatori, G., Kelsen, P.: Business Process Regulatory Compliance is Hard. IEEE Transactions on Services Computing 8(6), 958–970 (2015)
6. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
7. El Kharbili, M.: Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation. In: APCCM 2012, CRPIT, pp. 23–32. ACS (2012)
8. Governatori, G.: Business Process Compliance: An Abstract Normative Framework. IT – Information Technology 55(6), 231–238 (2013)
9. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems 14(2-3), 181–216 (2005)
10. Governatori, G.: The Regorous approach to process compliance. In: EVL-BP 2015, pp. 33–40. IEEE Press (2015)
11. Governatori, G.: Thou Shalt is not You Will. In: Atkinson, K. (ed.) ICAIL 2015, pp. 63–68. ACM, New York (2015)
12. Governatori, G., Hashmi, M., Lam, H.-P., Villata, S., Palmirani, M.: Semantic Business Process Compliance Checking Using LegalRuleML. In: Blomqbvist, E., Ciancarini, P., Poggi, F., Vitali, F. (eds.) EKAW 2016, LNAI, vol. 10024, pp. 746–761. Springer, Heidelberg (2016)
13. Governatori, G., Milosevic, Z.: Dealing with Contract Violations: Formalism and Domain Specific Language. In: EDOC 2005, pp. 46–57. IEEE Computer Society (2005)
14. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing Strong and Weak Permissions in Defeasible Logic. Journal of Philosophical Logic 42(6), 799–829 (2013)
15. Governatori, G., Rotolo, A.: A Conceptually Rich Model of Business Process Compliance. In: Link, S., Ghose, A. (eds.) APCCM 2010, CRPIT, vol. 110, pp. 3–12. ACS (2010)
16. Governatori, G., Rotolo, A.: Logic of Violations: A Gentzen System for Reasoning with Contrary-To-Duty Obligations. Australasian Journal of Logic 4, 193–215 (2006)

17. Governatori, G., Sadiq, S.: The Journey to Business Process Compliance. In: Cardoso, J., van der Aalst, W. (eds.) Handbook of Research on BPM, pp. 426–454. IGI Global (2009)
18. Governatori, G., Shek, S.: Regorous: A Business Process Compliance Checker. In: Francesconi, E., Verheij, B. (eds.) ICAIL 2013, pp. 245–246. ACM, New York (2013)
19. Hashmi, M., Governatori, G.: Norms Modeling Constructs of Business Process Compliance Management Frameworks: A Conceptual Evaluation. Artificial Intelligence and Law (2017)
20. Hashmi, M., Governatori, G., Lam, H.-P., Wynn, M.T.: Are We Done with Business Process Compliance: State-of-the-Art and Challenges Ahead. Knowledge and Information Systems (2018)
21. Hashmi, M., Governatori, G., Wynn, M.T.: Business Process Data Compliance. In: Bikakis, A., Giurca, A. (eds.) RuleML 2012, LNCS, vol. 7438, pp. 32–46. Springer, Heidelberg (2012)
22. Hashmi, M., Governatori, G., Wynn, M.T.: Normative Requirements for Regulatory Compliance: An Abstract Formal Framework. Information Systems Frontiers 18(3), 429–455 (2016)
23. Islam, M.B., Governatori, G.: Exposing the RuleRS: a rule-based architecture for decision support systems. Artificial Intelligence and Law (forthcoming)
24. Lam, H.-P., Governatori, G.: The Making of SPINdle. In: Governatori, G., Hall, J., Paschke, A. (eds.) RuleML 2009, LNCS, vol. 5858, pp. 315–322. Springer, Heidelberg (2009)
25. Sadiq, S., Governatori, G.: Managing Regulatory Compliance in Business Processes. In: vom Brocke, J., Rosemann, M. (eds.) Handbook of Business Process Management 2nd edition, International Handbooks on Information Systems, vol. 2, pp. 265–288. Springer, Berlin-Heidelberg (2015)
26. Sadiq, S., Governatori, G., Naimiri, K.: Modelling of Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007, LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)