A Methodological Evaluation of Business Process Compliance Management Frameworks

Mustafa Hashmi[1,2] and Guido Governatori[1,2]

[1] NICTA[†], Queensland Research Laboratory, 2 George St. Brisbane Australia
{mustafa.hashmi,guido.governatori}@nicta.com.au,
[2] Queensland University of Technology (QUT) Brisbane, Australia

Abstract. Existing compliance management frameworks (CMFs) offer a multitude of compliance management functionalities for the modeling of specific norms for specific domain and compliance checking of normative requirements. This makes difficult for enterprises to decide on a framework suitable for their compliance requirements. Making a decision on the suitability requires a deep understanding of the functionalities of a framework. Gaining such an understanding is a difficult task which, in turn, requires specialised tools and methodologies for evaluation. Current compliance research lacks such tools and methodologies for evaluating CMFs. This paper reports a methodological evaluation of existing CMFs based on pre-defined evaluation criteria. Our evaluation highlights what existing CMFs can offer, and what they cannot. Also, it underpins various open questions and discusses the challenges in this direction.

Key words: Business Processes, Compliance management, Compliance management frameworks

1 Introduction

The demand for reporting compliance puts pressure on enterprises to streamline their processes within the defined limits for better transparency and effective control over their operations. Essentially, compliance is an enterprise's ability to meet all the governing regulations enforced on its business operations. This demand has become even stronger after the fall of big corporate names like Enron, American International Group (AIG) which resulted in, due to non-adherence to regulations, the emergence of regulatory acts e.g., Sarbanex-Oxley Act, BASEL-III etc. These acts place restrictions and provide guidelines for enterprises on how to perform their business operations to stay compliant, and impose severe financial and criminal penalties otherwise.

Business processes provide enterprises an abstract view of the state of the affairs on how they are achieving business objectives, and implement regulatory policies governing their business operations. That is why enterprises use business processes to verify the effectiveness of regulatory laws and policy controls. Currently, enterprises employ a number of business process compliance checking strategies: *modeling-time* where the

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

The First Asia-Pacific Conference on Business Process Management (AP-BPM 2013),

analysts verify the non-compliant behavior before a process can be implemented, while processes are continuously monitored at *execution-time*, and logs are audited *after-execution* for any non-compliant patterns, see [13] for more details.

To support these strategies, a large body of compliance management frameworks (CMFs) has emerged, see [3] for a list of existing approaches. Each of these frameworks bears specific functional and operational capabilities, supports specific compliance requirements and domains, and claims to be a complete compliance solution. Such a multitude creates confusion for enterprises to decide on the suitability, and accordingly generate their compliance requirements. Deciding on the suitability requires a very careful and deep understanding on various features of a CMF which, in turns, is a difficult task and requires specific tools and methodologies. So far no accepted methodologies and tools exist that can be used to evaluate various features of CMFs. We address this issue and report a methodological evaluation of the selected CMFs on how they achieve compliance. The specific questions of this paper are: *how compliance is secured*, and *whether all types of norms are supported*. Addressing these questions will provide better understanding on the various functionalities of CMFs, also it identifies issues related to the normative requirements modeling and shortcoming of existing CMFs. Therefore, a diverse community may be benefited from this evaluation to obviate the deficiencies this evaluation may highlight.

The rest of the paper is structured as follows: Section 2 describes our research approach and a detailed discussion on the normative requirements followed by a comprehensive evaluation of the selected CMFs in Section 3. A short discussion on the conducted evaluations is presented in Section 4. Section 5 discusses the related work in the problem domain and gives some pointers on the future work to conclude the paper.

2 Research Approach

In this section, we present our research approach to conduct this evaluation. We adopted a systematic case study based shallow evaluation strategy which allowed us to start the evaluation with minimal information available on the CMFs. We followed a three steps structured approach where, we first defined the evaluation objectives and criteria, and then selected frameworks based on pre-defined criteria:

Evaluation Objectives: Our objective is to examine the conceptual foundations of existing CMFs. We specifically look at the conceptual approach a framework proposes to secure compliance, and the support for the normative requirements: more specifically what constructs are provided for modeling the norms. In addition, how the norms are linked to business processes for compliance checking.

Evaluation Criteria: We determined a three steps selection criteria to identify representative frameworks for this evaluation (1) *level of compliance management:* this criterion describes the level of support a framework provides. We only selected CMFs which provide full compliance management support and did not consider those merely provides a compliance checking algorithm or a modeling language, (2) *requirements modeling:* this criterion allows examining how frameworks model different types of compliance requirements, and using which formal logic. Essentially, this criterion is used to identify the modeling constructs for a specific obli-

gation type proposed in a framework. For this purpose, we provide a classification of normative requirements which has been obtained in a systematic and exhaustive way by considering the aspect of validity of obligations or prohibitions and what it means to violate them and what happens after they have been violated. (3) *requirements linking:* this criterion allows identifying how different frameworks link the compliance requirements with business process models for compliance checking.

Sample Frameworks Collection: Although we reviewed and analysed several CMFs, we abstained from doing a systematic literature survey as in [6] rather we selected frameworks based on expert discussions, and mostly cited in literature. In addition, we also considered the evaluation criteria while selecting the evaluated frameworks. We believe the selected frameworks are best suited for our evaluation according to the aforementioned criteria.

2.1 Normative Requirements

In the legal context, norms are meant to control the behaviour of their subjects and define whether something is legal or not. Typically norms describe their application conditions and the legal effects they produce when applied. [9] provides a comprehensive list of normative effects. From a compliance perspective the normative effects of interest are the deontic effects (a.k.a normative positions). The basic deontic effects are: *obligation, prohibition,* and *permission*[1]. Consider the definitions of the basic deontic effects as defined by the OASIS LegalRuleML working group[2].

Obligation: A situation, an act, or actions to which a bearer is legally bound, and if it is not achieved or performed results in a violation.

Prohibition: A situation, an act, or actions which a bearer should avoid, and if it is achieved or performed results in a violation.

Permission: Something is permitted if the obligations or prohibitions to the contrary do not hold.

Obligations and prohibitions act as constraints restricting the operations of business processes. What makes them different from other types of constraints is that they can be violated, but a violation does not mean inconsistency within the process with the consequent termination of or impossibility to continue the process. Also, it is common that violations can be compensated for, and processes with compensated violations are still compliant [13]. For example, usually contracts contain compensatory clauses specifying penalties and other sanctions to counter the violation of an obligatory clause [10]. However, not all violations are compensable, and uncompensated violations mean that a process is not compliant. Permissions are a special case of deontic effects and cannot be violated. Hence, they have no explicit role in compliance; rather they can be used to define that there are no obligations or prohibitions to the contrary, or to derive other deontic effects. Legal reasoning and legal theory typically assume a strong relationship between obligations and prohibitions as they are dual of each other: the *prohibition* of A is the obligation of $\neg A$, or if A is *obligatory* then $\neg A$ is prohibited [19].

[1] There are other deontic effects, but these can be derived from the basic ones, see [19].

[2] The OASIS LegalRuleML glossary is available at: `http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/48435/Glossary.doc`

Compliance aims at verifying whether a business process fulfills a set of obligations or not. For such a verification the first step is to determine whether and when an obligation is in force. Hence, an understanding on the lifetime of an obligation and related implications on the activities in a process is particularly important. This raises the question how long does an obligation hold for, and based on this which are the different conditions to fulfill the obligation. A norm can specify that an obligation is in force at a particular point of time, or more often indicates when it enters into force. An obligation remains active until it is terminated or removed. Accordingly, we will speak of a *punctual obligation* in the first case, and *persistent obligation* in the second.

In case of persistent obligations we can ask if we have to obey the obligation conditions for all instants in the interval in which it is in force, *maintenance obligations*, or whether meeting those conditions at least once is enough, *achievement obligations*. Furthermore, in case of an achievement obligation whether the obligation condition can be met even before the obligation is actually in force. If this is the case, then the obligation is *preemptive*, otherwise we have a *non-preemptive obligation*.

Termination of obligations is an important aspect of norms. Norms can specify the interval in which an obligation is in force. As was previously discussed that the violation of obligation conditions is what differentiates obligations from other constraints. The question is how the violation effects the obligation? Does the violation terminates the obligation or we still need to meet the conditions of a violated obligation? If we do –the obligation persists even after a violation –we speak of a *perdurant obligation*, and if not –then we have a *non-perdurant obligation*.

From the discussion above it is clear that different types of obligations have different compliance requirements. Hence, a '*one size fits all*' approach is far from being satisfactory from a conceptual point of view, and also a CMF not representing the various nuances of obligation is not conceptually sound, and it might not be suitable to provide any certification of compliance acceptable to accredited certifying organisations.

3 Evaluation of Frameworks

3.1 PENELOPE

PENELOPE (*Process Entailment from Elicitation of Obligations and Permissions* [8]) is a declarative language that captures obligation and permission constraints imposed on business processes by business policies. Aiming at providing a design-time compliance verification, the language uses an algorithm that progressively generates the *state space* and *control-flow* of a business process. The state space in a PENELOPE generated process is the set of obligations and permissions that are active at a particular state. The interaction between the generated process models flows from state to state, and all the states are enumerated until no obligation or permission holds at a state or if there is a violation which cannot be repaired. Once all the states are computed, the algorithm draws the BPMN model for a role involved in the business interaction. The tasks of the process are drawn whenever an obligation set contains all obligations fulfilled by a role in the activity. PENELOPE allows for the modeling of interaction between all involved partners and any violations from a third partner are represented as time out events in the

generated BPMN model. In addition, errors and end events are drawn if there is a violation of an obligation or permission by a role in a state. With the designed compliant process models various inconsistencies e.g. deontic conflicts can be identified.

The deontic assignments in the PENELOPE are modeled using event-calculus that provides a rich semantics to reason about the normative requirements. However, currently PENELOPE can only support achievement obligations and permissions while no other obligations types are explicitly supported as shown in Table 1. PENELOPE can model achievement obligations because they permit to explicitly define deadlines in the form of precedence rules. Prohibitions are not considered under close-world assumption (CWA) to avoid the anomalies that might occur because of incomplete knowledge about all the parties involved in the business interaction.

Violations in PENELOPE can only occur in the form of deadlock situations or temporal conflicts. Deontic conflicts cannot occur in PENELOPE generated BPMN models because the framework does not consider prohibitions or waived obligations. Moreover, no support for compensation obligation is provided because PENELOPE does not offer any mechanism to handle violations and this is left to the process modelers.

3.2 Process Compliance Language (PCL)

PCL (Process Compliance Language) [11] is a formal framework based on defeasible and deontic logic. It provides a conceptually rich formal foundation to model norms, and is able to efficiently capture the intuition of almost all types of normative requirements. Norms are modeled in the form of PCL rules for which the framework provides rich semantics. The state variables and the tasks in a process are represented by a set of propositional literals. $\neg$ negation, $\otimes$ non-boolean connective operator (for violations chains), and deontic operators representing obligations and permissions are used to construct the logic formulas called *PCL specifications*. The tasks in business processes are annotated with PCL specifications that are either provided by the domain experts or automatically extracted from the schemas of the databases or data sources linked to the processes [14]. These annotations are used to analyse whether the behavior of an execution path is consistent with the annotated specifications. For this purpose, a three-step algorithm is used in which first the process graph is traversed to find the set of effects for all tasks. These effects are then used to determine the norms in force for the tasks. The effects of the tasks and pertaining obligations are then compared in the last step to find any divergent behavior. The compliance of the norms is reported as full, partial, or not compliant by the algorithm.

The rich combination of defeasible and deontic logic allows PCL to model almost all types of obligations as depicted in Table 1 and other aspects of normative reasoning. This is because the use of two logics where deontic logic provides the support to model obligation's violations and chains of reparation, while the issue of partial information and inconsistent prescription is handled by defeasible logic [12]. To model the fundamental obligations PCL provides three modeling constructs: punctual (O^p), maintenance (O^m), and achievement (O^a), and an operator to express the orthogonal distinction between the classes of achievement obligations e.g., *premeptive obligations*.

Violations and obligations arising from the violations are major concerns in CMFs; PCL provides effective management of the violations and their compensations. For this

purpose PCL defines a special contrary-to-duty non-boolean $\otimes$ connective that is used to create reparation chains for handling multiple violations of obligations. As discussed in Section 2.1 some obligations may perdure and remain in force no matter how many times they may have been violated, but currently PCL cannot handle such obligations.

3.3 DECLARE

Declare [5] is a prominent framework for *run-time* verification of constraint-based declarative models. A declarative model describes *what a model does* by specifying the business constraints as rules that *should not be violated*. The business knowledge in Declare is defined in terms of constraints using ConDec (Constraint Declarative [18][3]), a language which provides graphical notations to model the flows of business interactions. Declare models (also templates) are enacted by a workflow engine that is used to verify the compliant interaction between the tasks in the model. The framework includes two types of constraints i.e., *mandatory* and *optional* constraints on the process models. In a Declare model, a process instance can only be active when there is not violation of the mandatory constraints and all the constraints are fully satisfied at the end of the execution of the instance. The verification results of each constraint of an active instance are expressed as *satisfied*, *temporarily violated*, and *violated*. In case all the constraints are satisfied the activities are not executed any further, but if there is a violation state no possible further execution would be allowed to satisfy the constraints. Accordingly, in the temporarily violated state the constraints are not satisfied, but there would be a possibility to satisfy the constraints.

Business constraints (norms) in the Declare framework are modeled by means of Declare expressions which are grouped as existence, relations, choice and negative constraints. The majority of these constraints are used to express obligations while the negative constraints express prohibitions. These constraints correspond to LTL expressions that provide the semantics to the Declare graphical notations. Currently, only achievement obligations and prohibitions can be modeled in the Declare Model, while no other norms types can be explicitly modeled, see Table 1. Since achievement obligation defines deadlines and the obligation condition must be true for at least once, the support for such obligations is only available because the tasks in the Declare model with such constraints will be performed in some future time. However, persistence and preemptiveness of obligations cannot be expressed. Constraints expressing maintenance obligation, on the other hand, can be problematic in Declare because the obligation conditions must hold in all instances throughout the execution of the process. There might be some situations when the applicable maintenance obligation constraints might not be present thus there will be deadlock in the course of interaction between the tasks. Declare is able to identify conflicts among constraints in the model; it does not provide any support to handle violations because of the lack of the declarative nature of the LTL and the non-deterministic behavior of the process models. Hence, in case of a violation the interaction between the tasks in the Declare model will be stopped and no further activ-

[3] From Nov 2012, the name of ConDec language has been changed to Declare see `http://www.win.tue.nl/declare/2011/11/declare-renaming/`.

ity can be performed. Accordingly, it is not possible to express permissions and so are compensations and perdurant obligations.

3.4 BPMN-Q

BPMN-Q (Business Process Modeling Notation-Query [1]) is a query based automated compliance checking framework capable to answer YES/NO type answers to query questions. The framework can model control-flow, data flow and conditional flow related compliance rules using visual patterns. These visual patterns are translated into LTL formulas for checking the structural compliance of a processes model. The framework adopts a systematic approach to generate the patterns of compliance rules in the form of query templates. These templates are used to identify the set of process models subject to compliance checking in the process repository. Compliance checking is carried out in several steps. First, BPMN-Q sub-graphs are extracted from the process repository using temporal query templates. The query processor only extracts processes that structurally match the query template. These sub-graphs are then reduced by eliminating irrelevant activities and gateways, and translated into a Petri Net model to generate the state space. Alongside the state space generation, BPMN-Q queries are translated into LTL formulas which are then fed into a model checker together with the generated state space. In turn, the model checker yields YES/NO to indicate whether the extracted process models comply with the query templates.

The framework uses a visual language BPMN-Q to express various types of compliance rules. The language provides visual notations similar to the standard BPMN notations. These visual notations provide the constructs to model compliance rules. Currently, the framework is able to handle almost the same types of obligations as the Declare framework with the exception of maintenance obligations (cf. Table 1). Maintenance obligations are expressed using the *global space presence* pattern which enables the execution of an activity that is required in all process instances.

In BPMN-Q no conceptual or formal constructs have been provided that can expressively model permissions. Whereas prohibitions are represented by global space absence to prevent the execution of some activities. Unlike the Declare framework, BPMN-Q is able to handle violations for which a violation handling approach has been discussed in [2]. Finally, compensations and perdurant obligations are not supported because of the use of LTL as underlying formalism to model compliance rules.

3.5 SEAFLOWS

SeaFlows [17] is a CMF for the verification of semantics constraints. It incorporates a graphical language providing primitives to capture process related complex business rules. These compliance rules are modeled in the form of *first-order logic* predicates equivalent and instantiable to compliance rules graphs (CRG). SeaFlow employs a structural compliance checking strategy for the verification of compliance rules where node relations are verified against the imposed constraints. The verification is done in three steps: in the first step a set of structural templates based on the queries on the relations of nodes in the process models is automatically derived. Then, the process

model is checked against the derived templates to detect any non-compliant structural templates. The queried templates are then aggregated and fed into the SeaFlows compliance module for further compliance report in the last step. The compliance results are based on the execution of traces of the process models where a process model is fully compliant when all the activities in the trace comply with the instantiated rule. Whereas a No is returned to indicate rule violations when no activity in the execution trace satisfies the rules.

To model compliance rules, the SeaFlows framework adopts a compositional graph-based modeling formalism allowing for the modeling of the typical antecedent-consequencet structure of rules. These graphs serve as placeholder for the first order logic representation of the relevant rules. Although SeaFlows is able to model achievement obligations which stipulate the occurrence of some event in future time by means of occurrence pattern, the framework is not able to capture other modalities e.g., punctual, maintenance, permissions, and compensation as depicted in Table 1. Moreover, compensations and perdurant obligations arising from the violation of the primary obligations cannot be modeled because first-order logic is not suitable to reason about the normative requirements [16].

4 Discussion

We have evaluated five CMFs using a set of pre-defined evaluation criteria. This methodological evaluation examined the salient features of CMFs, and *what is lacking* in terms of technical support in compliance domain especially for the modeling of normative requirements. The evaluation results are shown in Table 1, where the available support for modeling norms is indicated with '$+$', not supported with '$-$'.

It is clearly evident that only a fraction of normative requirements are supported as none of the evaluated CMFs is capable of supporting all types of norms. For example, PENELOPE is only able to support obligations and permissions. It is unable to model other obligation modalities, and violations because Event Calculus is not suitable for reasoning of legal constraints. Contrary to that, PCL supports almost all types of norms because of the non-monotonic characteristics of the formal logic it uses. The combination of defeasible and deontic logic allows PCL to provide reasoning for deontic modalities and violations especially for temporally varying obligations, e.g., achievement obligations and their persistence over time. As can be seen, PCL does not support perdurance obligation. DECLARE and BPMN-Q are LTL based frameworks, and only address 'structural compliance' where the tasks are defined by the constraint models. These frameworks cannot capture the intuition of all types of obligations, violations, and their compensations. DECLARE can only support achievement obligations and prohibitions while BPMN-Q can support achievement, maintenance, and prohibitions only. Generally it is highly desirable that a formal language is expressive enough to cover most of the properties and properties of the environment of the unit under verification. In addition, it should also support the complex properties from simpler ones, but temporal logic lacks such support because it has no conceptual relative correspondence to the legal domain, thus cannot expressively model the properties of the norms. The SeaFlows framework, on the other hand, can only support achievement, prohibitions

Table 1. Normative Requirements Support in Existing CMFs

Framework	Obligations					Permissions	Prohibitions	Violations
	Punctual	Achievement	Maintenance	Compensation	Perdurant			
PENELOPE	–	+	–	–	–	+	–	–
PCL	+	+	+	+	–	+	+	+
DECLARE	–	+	–	–	–	–	+	–
BPMN-Q	–	+	+	–	–	–	+	+
SEAFLOWS	–	+	–	–	–	–	+	+

and violations while other obligations modalities cannot be modeled. Because first order logic is not suitable to reasoning about the normative requirements as it does not provide operators to represent the properties of norms.

5 Conclusions

We presented a methodological evaluation of some existing CMFs using a shallow, but sound methodology where we examined the conceptual foundations under pre-defined evaluation criteria. Specifically we looked at the conceptual approaches existing CMFs use to deal with the normative requirements related to the regulatory compliance.

[3] offers a literature survey based on the generalisability and applicability of business process compliance frameworks. Their evaluation is based on the reported implementation results from the surveyed frameworks, while [6] compares the functional and non-functional capabilities of regulatory compliance management (RCM) solutions from a BPM perspective using a large set of evaluation criteria. Similarly, [4] studies various frameworks using a four point criteria including the study of modeling languages that are used to model business processes and rules. [7] conducted a comparative analysis of formal frameworks used to model the compliance rules, and [20] investigates the practices of regulation analysis and the approaches aiming to achieve and maintain regulatory compliance of given regulation from an Information Systems and services perspective. Our evaluation is complementary and different from these studies because we primarily evaluated existing CMFs to examine what they can do in terms of providing round-up compliance, and what constructs they provide to model different types of normative requirements. In addition, we also examined whether existing CMFs can provide reasoning support for all types of norms or not.

Our evaluation results portray somewhat a *bleak picture* when it comes to see how existing frameworks represent the legal knowledge for compliance checking because none is able to support all types of norms. Primarily this is because of the formal language each framework uses to model the norms. This highlights an exigent need for new compliance rules modeling languages with sound theoretical and formal foundations to effectively model and faithfully represent the legal knowledge, thus would increase the effectiveness of CMFs. We plan to conduct the formal semantics evaluation of these frameworks where we will examine the modeling behavior and constructs provided, and their correspondence to a specific modeling language. For this purpose, we have designed a formal framework in which we have provided detailed ontology and formal semantics for each of the normative requirements described in Section 2.1, [15]. The

planned evaluation will allow us to examine the expressive power of the compliance rules modeling languages, and to identify what is required in this direction?

References

1. A. Ahmed, G. Decker, and M. Weske. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In *BPM*, LNCS, pages 326–341. Springer, 2008.
2. A. Awad and M. Weske. Visualization of Compliance Violation in Business Process Models. In *5th Workshop on Business Process Intelligence BPI*, volume 9, pages 182–193, 2009.
3. J. Becker, P. Delfmann, M. Eggert, and S. Schwittay. 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.
4. C. Cabanillas, M. Resinas, and A. Ruiz-Cortes. Hints on how to face business process compliance. In *III Taller de Procesos de Negocio e Ingenieria de Servicios PNIS10 in JISBD10*, volume 4, pages 26–32, 2010.
5. DECLARE. Declarative process models, http://www.win.tue.nl/declare/.
6. M. El Kharbili. Business process regulatory compliance management solution frameworks: A comparative evaluation. In *APCCM 2012*, CRPIT 130, pages 23–32, 2012.
7. A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou. On the formal specification of regulatory compliance: a comparative analysis. In *Proceedings of ICSOC'10*, pages 27–38, 2011.
8. S. Goedertier and J. Vanthienen. Designing Compliant Business Processes with Obligations and Permissions. In *BPM Workshops*, volume 4103 of *LNCS*, pages 5–14. Springer, 2006.
9. T. F. Gordon, G. Governatori, and A. Rotolo. Rules and norms: Requirements for rule interchange languages in the legal domain. In *RuleML 2009*, LNCS 5858, pages 282–296. Springer, 2009.
10. G. Governatori. Representing Business Contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.
11. G. Governatori and A. Rotolo. A Conceptually Rich Model of Business Process Compliance. In *Proceedings of APCCM '10*, volume 110, pages 3–12. ACS, Inc. Australia, 2010.
12. G. Governatori and A. Rotolo. Norm compliance in business process modeling. In *RuleML 2010: 4th International Web Rule Symposium*, pages 194–209. Springer, 2010.
13. G. Governatori and S. Sadiq. The Journey to Business Process Compliance. In *Handbook of Research on Business Process Management*, pages 426–454. IGI Global, 2009.
14. M. Hashmi, G. Governatori, and M. T. Wynn. Business Process Data Compliance. In A. Bikakis and A. Giurca, editors, *LNCS*, volume 7438, pages 32–46. Springer, 2012.
15. M. Hashmi, G. Governatori, and M. T. Wynn. Normative Requirements for Buisness Process Compliance. Technical report, Queensland University of Technology, Brisbane, Australia, 2013.
16. H. Herrestad. Norms and formalization. In *ICAIL'91*, pages 175–184. ACM, 1991.
17. L. T. Ly, D. Knuplesch, S. Rinderle-Ma, K. Goeser, M. Reichert, and P. Dadam. Seaflows toolset - compliance verification made easy. In *CAiSE'10 Demos*, June 2010.
18. M. Pesic and W. van der Aalst. A declarative approach for flexible business processes management. In *BPM Workshops*, volume 4103 of *LNCS*, pages 169–180. Springer, 2006.
19. G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.
20. S. Turki and M. Bjekovic-Obradovic. Compliance in e-government service engineering: State-of-the-art. In *Exploring Services Science*, LNBIP, pages 270–275. Springer, 2010.