it – Information Technology
55 (2013) 6, 231–238

Business Process Compliance: An Abstract Normative Framework

Guido Governatori

Abstract
In this paper we propose an abstract framework to model the deontic notions relevant for business process compliance. In particular, we provide a comprehensive classification of the obligation types relevant for modelling whether a process is compliant, and we describe their semantics in terms of execution traces.

Keywords
Regulatory Compliance, Business Processes, Deontic Logic

NICTA, Queensland Research Laboratory

1. Introduction

The study of IT techniques to support compliance is gaining momentum in the area of Enterprise Information Systems. The number and complexity of compliance initiatives and frameworks is growing and more and more businesses are required to provide compliance certification by such frameworks. In the past few years several research approaches have been proposed (see [2, 4] for comprehensive surveys).

Regulatory compliance is defined as the set of actives and policies in place in an enterprise to ensure the business activities required to achieve the business goals of the company comply with the relevant normative requirements. Here *normative requirements* must be understood with a very broad interpretation. They include statutory laws, regulations, industry codes, standards, internal policies, ... Despite the differences, they share a common aspect: they describe the obligations, prohibitions and permissions an organization is subject to in its day to day business. *Business Process Compliance* has been defined as a relationship between the formal specifications of a process and the formal representation of the regulatory frameworks relevant of the process [13, 20]. More specifically, a process is compliant if its formal specifications are compatible with the formal representation of the normative requirements, meaning that properly executing a process does not results in violations of the normative requirements.

The aim of this paper is not to provide a yet another system for compliance, but a conceptual abstract framework, independent of any language, to model normative requirements. The resulting framework can be used for several purposes: a first possible use is to study formal properties of business process compliance, for example, the computational complexity of the problem of determining whether a given business process complies with a specific regulation [3], and to possibly identify tractable subclasses; a second application is to provide a precise ground to compare different approaches to business process compliance, to identify gaps in the modelling of the normative requirement, and to assess their suitability to model business process compliance [15].

For the purpose of this paper a process will be understood as a set of sequences of tasks. In addition every task has associated to it a set of effects where an effect is just a formula of the underlying language.

For the representation of the normative requirements, we propose a classification of the various normative positions (i.e., obligations, permissions, prohibitions) and we provide further refinements for them along various dimensions (e.g., temporal, compensability, perdurance). For each class we provide the semantics and examples, extracted from actual acts, codes and regulations, illustrating the particular types of normative positions.

2. Business Process Modelling

A business process model is a self-contained, temporal and logical order in which a set of activities are executed to achieve a business goal. Typically a process model describes what needs to be done and when (control flow), who is going to do what (resources), and on what it is working on (data). Many different formalisms (Petri-Nets, Process algebras, ...) and notations (BPMN, YAWL, EPC, ...) have been proposed to represent business process models. Besides the difference in notation, purposes, and expressive power, business process languages typically contain the following minimal set of elements:

- tasks
- connectors
- events

where 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, in this paper, we limit ourselves to the *start* and the *end* events signalling, respectively, when a process begins and when a process terminates. The combination

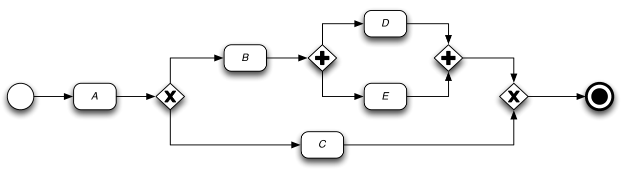


Figure 1. Example of a business process model in standard BPMN notation

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 respecting the order given by the connectors.

Consider the process in Figure 1, in standard BPMN notation, where we have a task A followed by an xor split. In the xor split in one of the branches we have task B followed by the and-split of a branch with task D, and a brach consisting of only task E. The second branch of the xor-split has only one task: C. The traces corresponding to the process are $\langle A,C\rangle$, $\langle A,B,D,E\rangle$ and $\langle A,B,E,D\rangle$. Given a process P we will use $\mathcal{T}_P = \{t_1,t_2,\dots\}$ to denote the set of traces of P.

Compliance is not only about the tasks that an organisation has to perform to achieve its business goals, but it is concerned also on their effects (i.e., how the activities in the tasks change the environment in which they operate), and the artefacts produced by the tasks (for example, the data resulting from executing a task or modified by the task) [16]. To capture this aspect [21] 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. Accordingly, we extend the notion of trace. To this end we introduce the function

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

where $\mathcal{L}$ is the set of formulas of the language used to model the annotations. The meaning of the function *State* is to identify the propositions that hold (are evaluated as true) after the execution of a task in a trace.

Let us illustrate the above ideas with the help of an example.

Example 1

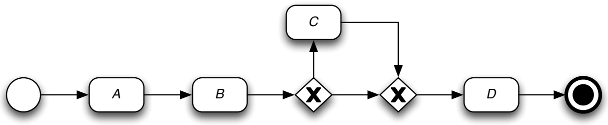


Figure 2. Business Process of Example 1

Consider the business process in Figure 2 where task A is "turn the light on", task B is "check if the glass is empty", task C is "fill the glass with water", and task D is "turn the glass upside down". The annotations we consider are built from a language with the following two propositions: p meaning "the light is on" and q meaning "the glass is full". The process has two traces: $t_1 = \langle A,B,D\rangle$ and $t_2 = \langle A,B,C,D\rangle$. In both traces after task A, the first task of both traces, we have that the light is on. This can be represented by $State(t_1,1) = \{p\}$ and $State(t_2,1) = \{p\}$. The second task, common to the two traces, is to check if the glass is empty. Here we have a difference. In case it is we have to fill it with water (task C) before turing the glass upside down (task D); otherwise we directly turn the glass upside down. This means that what we know after the second task depends on the trace we are examining. In trace t_1 the information is that the glass is full (i.e., not empty), which is then represented by $State(t_1,2) = \{p,q\}$, while in trace t_2 the glass is empty, formally $State(t_2,2) = \{p,\neg q\}$. After the next step in trace t_2 (task C filling the glass with water) we are in a state where the glass is empty, thus we have $State(t_2,3) = \{p,q\}$. The effects of the last task (task D turing the glass upside down) result in the glass be empty. Accordingly, we have $State(t_1,3) = State(t_2,4) = \{p,\neg q\}$.

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. However, a trace uniquely determines the sequence of states obtained by executing the trace. Thus, in what follows we use a trace to refer to a sequence of tasks, and the corresponding sequence of states.

3. Business Process Compliance

The set of traces of a given business process describes the behavior of the process insofar as it provides a description of all possible ways in which the process can be correctly executed. Accordingly, for the purpose of defining what it means for a process to be compliant, we will consider a process as the set of its traces.

Intuitively a process is compliant with a normative system[1] if it does not breach the normative system. Given that, in general, it is possible to perform a business process in many different ways, thus we can have two notions of compliance, namely:

(S1) A process is (fully) compliant with a normative system if it is impossible to violate the normative system while executing the process.

The intuition about the above condition is that no matter in which way the process is executed, its execution does not

[1]Here, by normative system we simply mean a set of norms, where a norm is a formula in the underlying (deontic) language. For a business process the normative system could vary from a particular regulation, to a specific statutory act, a set of best practices, a standard, simply a policy internal to an organisation or a combination of these types of prescriptive documents.

violate the normative system. For the second one we consider the case that there is an execution of the process that does not violate the norms.

(S2) A process is (partially) compliant with a normative system if it is possible to execute the process without violating the normative system.

Based on the above intuition we can give the following definition:

Definition 1 *Let $\mathcal{N}$ be a normative system.*

1. *A process P* fully complies *with $\mathcal{N}$ if and only if every trace $t \in \mathcal{T}_P$ complies with $\mathcal{N}$.*
2. *A process P* partially complies *with $\mathcal{N}$ if and only if there is a trace $t \in \mathcal{T}_P$ that complies with $\mathcal{N}$.*

Notice that in (S1) and (S2) compliance means "lack of violations" while in Definition 1 we had "comply with". For the purpose of this paper we will treat these two concepts as equivalent. More precisely they are related by the following definition.

Definition 2 *A trace t complies with a normative system $\mathcal{N} = \{n_1, n_2, \dots\}$ if and only if all norms in $\mathcal{N}$ have not been violated.*

In Section 4 we are going to introduce various types of norms. For each type we are going to describe its semantics in terms of what constitutes a violation of a norm of that type.

The possibility of a norm to be violated is what distinguish norms from other types of constraints. Then, given that violations are possible, one has to consider that violations can be compensated. Is a process where some norms have violated and compensated for compliant? To account for this possibility we introduce the distinction between *strong* and *weak* compliance. Strong compliance corresponds to Definition 2. Weak compliance is defined as follows:

Definition 3 *A trace t is weakly compliant with a normative system $\mathcal{N}$ if and only if every violated norm has been compensated for.*

Remark 1 *It is not the scope of this paper to describe how the sequences of states corresponding to the executions of a process are obtained. The task of specifying how the function State is implemented is left to specific compliance applications. For example one can use the update semantics approach [6] or using Event Calculus to model the inertia of effects from one task the next one as done in [7].*

4. Normative Requirements

The scope of norms is to regulate the behaviour of their subjects and to define what is legal and what is illegal. Norms typically describe the conditions under which they are applicable and the normative effects they produce when applied. A comprehensive list of normative effects is provided in [8]. In a compliance perspective, the normative effects of importance are the deontic effects (also called normative positions). The basic deontic effects are: *obligation*, *prohibition* and *permission.*[2]

Let us start by consider the basic definitions for such concepts:[3]

Obligation A situation, an act, or a course of action 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 a course of action which a bearer should avoid, and if it is achieved results in a violation.

Permission Something is permitted if the obligation or the prohibition to the contrary does not hold.

Obligations and prohibitions are constraints that limit the space of action of processes; the difference from other types of constraints is that they can be violated, and a violation does not imply an inconsistency within a process with the consequent termination of or impossibility to continue the business process. Furthermore, it is common that violations can be compensated for, and processes with compensated violations are still compliant (or weakly compliant) [10, 13]; for example contracts typically contain compensatory clauses specifying penalties and other sanctions triggered by breaches of other contract clauses [9]. Not all violations are compensable, and uncompensated violations means that a process is not compliant. Permissions cannot be violated, thus permissions do not play a direct role in compliance; they can be used to determine 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: the prohibition of A is the obligation of $\neg A$ (the opposite of A), and then if A is obligatory, then $\neg A$ is forbidden [22]. In this paper we will subscribe to this position, given that our focus here is not on how to determine what is prescribed by a set of norms and how to derive it. Accordingly, we can restrict our analysis to the notion of *obligation.*

Compliance means to identify whether a process violates or not a set of obligations. Thus, the first step is to determine whether and when an obligation is in force. Hence, an important aspect of the study of obligations is to understand the lifespan of an obligation and its implications on the activities carried out in a process. As we have alluded to above norms give the conditions of applicability of obligations. The question then is how long does an obligation hold for, and based on this there are different conditions to fulfill the obligation. We take a systematic approach to this issue. A norm can specify that an obligation is in force for a

[2] There are other deontic effects, but these can be derived from the basic ones, see [22].

[3] Here we consider the definition of such concepts given by the OASIS LegalRuleML working group. The OASIS LegalRuleML glossary is available at `http://www.oasis-open.org/apps/org/workgroup/legalruleml/download.php/48435/Glossary.doc`.

particular time point or, more often, a norm indicates when an obligation enters in force. An obligation remains in force until terminated or removed. Accordingly, in the first case we will speak of *punctual obligations* and in the second case of *persistent obligations*.

For persistent obligations we can ask if to fulfill an obligation we have to obey to it for all instants in the interval in which it is in force, *maintenance obligations*, or whether doing or achieving the content of the obligation at least once is enough to fulfill it, *achievement obligations*. For achievement obligations another aspect to consider is whether the obligation could be fulfilled even before the obligation is actually in force. If this is admitted, then we have a *preemptive obligation*, otherwise the obligation is *non-preemptive*.

The final aspect we want to touch upon in this section is the termination of obligations. Norms can specify the interval in which an obligation is in force. Previously, we discussed that what differentiates obligations and other constraints is that obligations can be violated. What are the effects of a violation on the obligation the violation violates? More precisely, does a violation terminate the violated obligation? Meaning, do we still have to comply with a violated obligation? If we do –the obligation persists after being violated– we speak of a *perdurant obligation*, if it does not, then we have a *non-perdurant obligation*.

It is worth noticing that the classification discussed above is exhaustive. It has been obtained in a systematic and comprehensive way when one considers the aspect of the validity of obligations –or prohibitions– (i.e., whether they persist after they enter in force or they are valid only for a specific time unit), and the effects of violations on them, namely: whether a violation can be compensated for, and whether an obligation persists after being violated. In the next section we will provide formal definitions for the notions introduced in this section and for each case we will show examples taken form statutory Acts and other legally binding documents.

5. Modelling Obligations

In this section we provide the formal definitions underpinning the notion of compliance. In particular we formally define the different types of obligations introduced in Section 4.

Definition 4 (Obligation in force) *Given a process P, and a trace $t \in \mathcal{T}_P$. We define a function*

$$Force\colon \mathcal{T}_P \times \mathbb{N} \mapsto 2^{\mathcal{L}}.$$

The function *Force* associates to each task in a trace a set of literals, where these literals represent the obligations in force for that combination of task and trace. These are among the obligations that the process has to fulfill to comply with a given normative framework. For example given a trace t, $Force(t,2) = \{p,q\}$ means that p and q are obligatory in the second task of trace t.

In the rest of the section we are going to give definitions specifying when the process has to fulfill the various obligations (depending on their type) to be deemed compliant.

Remark 2 *Similarly to Remark 1 we are not interested in the mechanisms that establish which obligations are in force and when. This is the scope of specific compliance applications or implementations.*

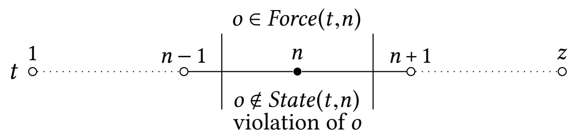
Definition 5 (Punctual Obligation) *Given a process p and a trace $t \in \mathcal{T}_P$, an obligation o is a* punctual obligation *in t if and only if $\exists n \in \mathbb{N}$ such that*

1. $o \notin Force(t,n-1)$,
2. $o \notin Force(t,n+1)$, *and*
3. $o \in Force(t,n)$.

The punctual obligation o is in force at n *in t.*

A punctual obligation o in force at n in t is violated *if and only if $o \notin State(t,n)$.*

The following diagram illustrates the definition above.



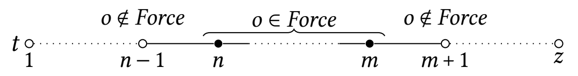
A punctual obligation is an obligation that is in force in one task of a trace (it might be the case that there are multiple instances in which the obligation is in force). The obligation is violated if what the obligation prescribes is not achieved in or done by the task, where this is represented by the literal not being in the set of literals associated to the task in the trace.

Definition 6 (Persistent Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is a* persistent obligation *in t if and only if $\exists n,m \in \mathbb{N}, n < m$ such that*

1. $o \notin Force(t,n-1)$,
2. $o \notin Force(t,m+1)$, *and*
3. $\forall k : n \leq k \leq m, o \in Force(t,k)$

The obligation o is in force between n and m.

A persistent obligation is an obligation in force in an interval of tasks in a process. The diagram below depicts the definition of when a persistent obligation o is in force between n and m.



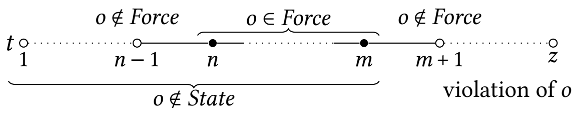
As we discussed before persistent obligations can be further classified as achievement and maintenance obligations. The difference between them is the conditions under which we can assert that the obligations have been violated.

Definition 7 (Achievement Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is an* achievement obligation *in t if and only if $\exists n,m \in \mathbb{N}, n < m$ such that o is a persistent obligation in force between n and m.*

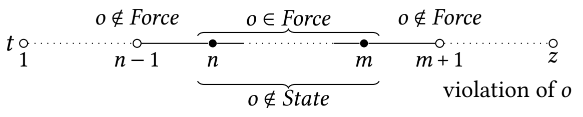
An achievement obligation o in force between n and m in t is violated *if and only if*

- *o is* preemptive *and* $\forall k\colon k \le m$, $o \notin State(t,k)$;
- *o is* non-preemptive *and* $\forall k\colon n \le k \le m$, $o \notin State(t,k)$.

An achievement obligation is in force in a contiguous set of tasks in a trace. The violation depends on whether we have a preemptive or a non-preemptive obligation. A preemptive obligation o is violated if no state before the last task in which o is in force has o in its annotations. The following diagram graphically represents the definition of non-preemptive achievement obligations.



For a non-preemptive obligation the set of states one has to consider to determine whether the obligation has been violated is restricted to those defined by the interval in which the obligation is in force; see the diagram below for a pictorial description of the case.



Example 2 *Australian Telecommunications Consumers Protection Code 2012 (TCPC 2012). Article 8.2.1.*
A Supplier must take the following actions to enable this outcome:

(a) ***Demonstrate fairness, courtesy, objectivity and efficiency:*** *Suppliers must demonstrate, fairness and courtesy, objectivity, and efficiency by:*
  (i) Acknowledging a Complaint:
    A. immediately where the Complaint is made in person or by telephone;
    B. within 2 Working Days of receipt where the Complaint is made by email; ....

The obligation to acknowledge a compliant made in person or by phone (8.2.1.a.i.A) is a punctual obligation, since it has to be done 'immediately' while receiving it (thus it can be one of the activities done in the task 'receive complaint'). 8.2.1.a.i.B on the other hand is an achievement obligation since the clause gives a deadline to achieve it. In addition it is a non-preemptive obligation. It is not possible to acknowledge a complaint before having it.

The next example exhibits a case of preemptive achievement obligation.

Example 3 *Australian National Consumer Credit Protection Act 2009. Schedule 1, Part 2, Section 20: Copy of contract for debtor.*

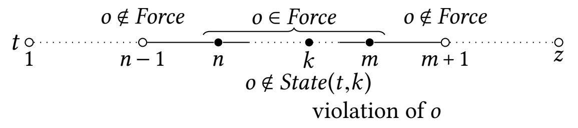
(1) If a contract document is to be signed by the debtor and returned to the credit provider, the credit provider must give the debtor a copy to keep.
(2) A credit provider must, not later than 14 days after a credit contract is made, give a copy of the contract in the form in which it was made to the debtor.
(3) Subsection (2) does not apply if the credit provider has previously given the debtor a copy of the contract document to keep.

Definition 8 (Maintenance Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is a* maintenance obligation *in t if and only if $\exists n,m \in \mathbb{N}, n < m$ such that o is a persistent obligation in force between n and m.*

A maintenance obligation o in force between n and m in t is violated *if and only if*

$$\exists k\colon n \le k \le m, o \notin State(t,k).$$

Similarly to an achievement obligation, a maintenance obligation is in force in an interval. The difference is that the obligation has to be complied with for all tasks in the interval, otherwise we have a violation. See the following diagram for a graphical illustration of the structure of maintenance obligations.



Example 4 *TCPC 2012. Article 8.2.1.*
A Supplier must take the following actions to enable this outcome:

(v) not taking Credit Management action in relation to a specified disputed amount that is the subject of an unresolved Complaint in circumstances where the Supplier is aware that the Complaint has not been Resolved to the satisfaction of the Consumer and is being investigated by the Supplier, the TIO or a relevant recognised third party;

In this example, as it is often the case, a maintenance obligation implements a prohibition. Specifically, it describes the prohibition to initiate a particular type of activity until either a particular event takes place or a state is reached.

The next three definitions are meant to capture the notion of compensation of a violation. The idea is that a compensation is a set of penalties or sanctions imposed on the violator, and fulfilling them makes amend for the violation. The first step is to define what a compensation is. A compensation is a set of obligations in force after a violation of an obligation (Definitions 9 and 10). Since the compensations are obligations themselves they can be violated, and they can be compensable as well, thus we need a recursive definition for the notion of compensated obligation (Definition 11).

Definition 9 (Compensation) *A* compensation *is a function Comp*: $\mathcal{L} \mapsto 2^{\mathcal{L}}$.

Definition 10 (Compensable Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is* compensable *in t if and only if*

1. *$Comp(o) \neq \emptyset$ and*
2. *$\forall o' \in Comp(o), \exists n \in \mathbb{N}\colon o' \in Force(t,n)$.*

Definition 11 (Compensated Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an obligation o is* compensated *in t if and only if it is violated and for every $o' \in Comp(o)$ either:*

1. *o' is not violated in t, or*
2. *o' is compensated in t.*

For a stricter notion, i.e., a compensated compensation does not amend the violation the compensation was meant to compensate, we can simply remove the recursive call, thus removing 2. from the above condition.

Compensations can be used for two purposes. The first is to specify alternative, less ideal outcomes. The second is to capture sanctions and penalties. Examples 5 and 6 below illustrate, respectively, these two usages.

Example 5 *TCPC 2012. Article 8.1.1.*
A Supplier must take the following actions to enable this outcome:

(a) ***Implement a process****: implement, operate and comply with a Complaint handling process that:*
   (vii) requires all Complaints to be:
      A. Resolved in an objective, efficient and fair manner; and
      B. escalated and managed under the Supplier's internal escalation process if requested by the Consumer or a former Customer.

Example 6 *YAWL Deed of Assignment, Clause 5.2.*[4]
Each Contributor indemnifies and will defend the Foundation against any claim, liability, loss, damages, cost and expenses suffered or incurred by the Foundation as a result of any breach of the warranties given by the Contributor under ***clause 5.1****.*

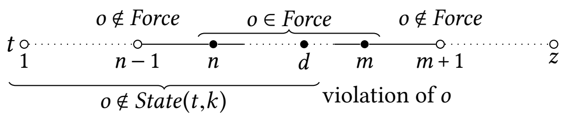
The final definition is that of perdurant obligation. The intuition behind it is that there is a deadline by when the obligation has to be fulfilled. If it is not fulfilled by the deadline then a violation is raised, but the obligation is still in force. Typically, the violation of a perdurant obligation triggers a penalty, thus if the perdurant obligation is not fulfilled in time, then the process has to account for the original obligation as well as the penalties associated with the violation.

Definition 12 (Perdurant Obligation) *Given a process P and a trace $t \in \mathcal{T}_P$, an achievement obligation o is a* perdurant obligation[5] *in t* with deadline *d if and only if o is in force between n and m and $n < d < m$.*

A perdurant obligation o with deadline d in force between n and m is violated *in t if and only if*

$$\forall j, j \leq d,\ o \notin State(t,d)$$

Also in this case we provide a diagram illustrating the notions just defined.



Consider again Example 2. Clauses TCPC 8.2.1.a.i.A and 8.2.1.a.i.B state what are the deadlines to acknowledge a complaint, but 8.2.1.a.i prescribes that complaints have to be acknowledged. Thus, if a complaint is not acknowledged within the prescribed time then either clause A or B are violated, but the supplier still has the obligation to acknowledge the complaint. Thus the obligation in clause (i) is a perdurant obligation.

6. ICT Approaches to Compliance

Based on the discussion in the previous section the problem of checking whether a business process is compliant against a regulation can be split in two components: (i) determining what obligations/prohibitions are in force, and when they are in force, and (ii) verifying whether executions of the business process satisfy the obligations and prohibitions in force.

Several frameworks have been prosed to address the issue of business process compliance. Here we consider some representative ones: MoBuCom [19], Compas [5], BPMN-Q [1], SeaFlows [18], PENELOPE [7] and FCL/PCL [9, 12, 11].

MoBuCom and Compas are based on Linear Temporal Logic (LTL) and BPMN-Q on Computational Tree Logic (CTL); mostly they just address "structural compliance" (i.e., that the tasks are executed in the relative order defined by a constraint model). The use of temporal logic implies that the model on which these tools are based on is not conceptual relative to the legal domain, and it fails to capture nuances of reasoning with normative constrains such as violations, different types of obligations, violations and their compensation. For example, obligations are represented by temporal operators. This raises the problem of how to represent the distinction between achievement and maintenance obligations. A possible solution is to use the always operator for maintenance and the sometimes operator for achievement, but this leaves no room for the concept of permission (the permission is dual of obligation, and always and sometimes are the dual of each other). In addition using temporal operators to model

[4]`http://www.yawlfoundation.org/files/YAWLDeedOfAssignmentTemplate.pdf`, retrieved on March 28, 2013.

[5]For space reasons, we show only the definition of perdurant obligations for preemptive achievement obligations. Similar notions can be defined for non-preemptive and maintenance obligations by simple adjustments of the definition.

obligations makes hard to capture data compliance [16], i.e., obligations that refer to literals in the same task.

SeaFlows is based on First-Order Logic, and it is well know that First-Order Logic is not suitable to capture normative reasoning, in particular in presence of violations [17]. Furthermore, First-Order Logic is not able to distinguish between obligations, prohibitions and permissions. These problems are also present in the approach of [6] based on clausal form and satisfiability.

PENELOPE is based on a combination of Event Calculus and Deontic Logic. Event Calculus is used to reason about when obligations enter in force, and when they cease to be effective. Thus it provides a native and conceptual support for the notion of obligations. Currently PENELOPE only supports achievement obligations and permissions while no other obligations types are explicitly supported (see [15] for a more detailed analysis).

PCL (Process Compliance Logic) combines Defeasible Logic (for the efficient and natural treatment of exceptions, which are a common feature in normative reasoning) and a deontic logic of violations. To the best of our knowledge PCL is the only compliance management framework supporting all the types of normative requirements presented in this paper (on this analysis see also [15]). Furthermore, PCL complies with the guidelines set up in [8] for a rule languages for the representation of legal knowledge and legal reasoning.

PCL and the abstract framework developed in this paper have been evaluated with an industry study reported in [14]. The study examined Section 8 of the 2012 Australian Telecommunication Customer Protection Code about complaint handling. The normative requirements specified in the Code were manually mapped in 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. Of the 176 rules 33 were used to capture definitions of terms used in the remaining rules. All types of normative requirement described in the paper were found in Section 8 of the code. The table below reports the types of deontic effects present in the PCL mapping, and for each type the table includes the number of distinct occurrences and, in parenthesis, the total number of instances (some effects can have different conditions under which they are in force).

Punctual Obligation		5	(5)
Achievement Obligation		90	(110)
	Preemptive	41	(46)
	Non preemptive	49	(64)
	Non perdurant	5	(7)
Maintenance Obligation		11	(13)
	Prohibition	7	(9)
	Non perdurant	1	(4)
Permission		9	(16)
Compensation		2	(2)

In addition the compliant handling processes of an industry partner operating in the sector were modelled and tested for compliance. The exercise was fruitful insofar as the industry partner was able to identify some non compliance issues with the novel code, and consequently was able to rectify them, and to generate compliance verifiable processes.

7. Conclusion

In this paper we presented an abstract framework to describe the key concepts of business process compliance. As far as we know this the first time that the problem of compliance has been give a precise and formal treatment taking into account the formalisation of the normative requirements. In particular we provided a comprehensive classification of obligations and their semantics in terms of the execution traces of a process. The proposed model is neutral from specific logics for reasoning with norms and process model formalisms. In addition for each type of obligation we provided examples taken from actual real life legal codes and legislative acts.

Acknowledgment
NICTA is funded by the Australian Government and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] A. Awad, M. Weidlich, and M. Weske. “Visually specifying compliance rules and explaining their violations for business processes”. *Journal of Visual Languages & Computing*. 22(1): 30–55, 2011.

[2] 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.

[3] S. Colombo Tosatto, G. Governatori, P. Kelsen, and L. van der Torre. *Business Process Compliance is Hard*. NICTA Technical Report, 2012.

[4] M. El Kharbili. “Business Process Regulatory Compliance Management Solution Frameworks: A Comparative Evaluation”. In *APCCM 2012*. CRPIT 130, pp. 23–32, 2012.

[5] A. Elgammal, O. Türetken, and W.-J. Van Den Heuvel. “Using patterns for the analysis and resolution of compliance violations”. *International Journal of Cooperative Information Systems*. 21(1): 31–54, 2012.

[6] A. Ghose and G. Koliadis. “Auditing Business Process Compliance”. In *ICSOC 2007*. LNCS 4749, pp. 169–180. Springer, 2007.

[7] S. Goedertier and J. Vanthienen. *Designing Compliant Business Processes with Obligations and Permissions*. Ed. by J. Eder and S. Dustdar. 2006, 2006.

[8] 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, pp. 282–296. Springer, 2009.

[9] G. Governatori. “Representing Business Contracts in RuleML”. *International Journal of Cooperative Information Systems*. 14(2-3): 181–216, 2005.

[10] G. Governatori and Z. Milosevic. “Dealing with contract violations: formalism and domain specific language”. In *EDOC 2005*, pp. 46–57. IEEE Computer Society, 2005.

[11] G. Governatori and A. Rotolo. “A Conceptually Rich Model of Business Process Compliance”. In *APCCM’10*. CRPIT 110, pp. 3–12, 2010.

[12] G. Governatori and A. Rotolo. “Norm Compliance in Business Process Modeling”. In *RuleML’10*. LNCS 6403, pp. 194–209. Springer, 2010.

[13] G. Governatori and S. Sadiq. “The Journey to Business Process Compliance”. In *Handbook of Research on Business Process Management*, pp. 426–454. IGI Global, 2009.

[14] G. Governatori and S. Shek. “Rule Based Business Process Compliance”. In *RuleML2012@ECAI Challenge*. CEUR Workshop Proceedings 874, article 5, 2012.

[15] M. Hashmi and G. Governatori. “A Methodological Evaluation of Business Process Compliance Management Frameworks”. In *AP-BPM 2013*. LNBIP 159, pp. 106–115. Springer, 2013.

[16] M. Hashmi, G. Governatori, and M.T. Wynn. “Business Process Data Compliance”. In *RuleML 2012*. LNCS 7438, pp. 32–46. Springer, 2012.

[17] H. Herrestad. “Norms and formalization”. In *ICAIL’91*, pp. 175–184. ACM, 1991.

[18] L.T. Ly, S. Rinderle-Ma, K. Göser, and P. Dadam. “On enabling integrated process compliance with semantic constraints in process management systems”. *Information Systems Frontiers*. 14(2): 195–219, 2012.

[19] F. Maggi, M. Montali, M. Westergaard, and W. van der Aalst. “Monitoring Business Constraints with Linear Temporal Logic: An Approach Based on Colored Automata”. In *BPM 2011*. LNCS 6896, pp. 132–147. Springer, 2011.

[20] S. Sadiq and G. Governatori. “Managing Regulatory Compliance in Business Processes”. In van Brocke, J., and M. Rosemann, eds. *Handbook of Business Process Management*. Vol. 2, pp. 157–173. Springer, 2010.

[21] S. Sadiq, G. Governatori, and K. Namiri. “Modeling Control Objectives for Business Process Compliance”. In *BPM 2007*. LNCS 4714, pp. 149–164. Springer, 2007.

[22] G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.