Normative Requirements for Business Process Compliance*

Mustafa Hashmi[1,2], Guido Governatori[1,2], and Moe Thandar Wynn[2,1]

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

Abstract. Norms regulate the behaviour of their subjects and define what is legal and what is illegal. Norms typically describe the conditions under which they are applicable and the normative effects as a result of their applications. On the other hand, process models specify how a business operation or service is to be carried out to achieve a desired outcome. Norms can have a significant impact on how business operations are conducted and they can apply to the whole or a part of a business process. For example, they may impose conditions on the different aspects of a process (e.g., perform tasks in a specific sequence (control-flow), at a specific time or within a certain time frame (temporal aspect), by specific people (resources)). We propose a framework that provides the formal semantics of the normative requirements for determining whether a business process complies with a normative document (where a normative document can be understood in a very broad sense, ranging from internal policies to best practice policies, to statutory acts). We also present a classification of normal requirements based on the notion of different types of obligations and the effects of violating these obligations.

Key words: Normative requirements, regulatory compliance, business process compliance

1 Introduction

Due to ever increasing pressure and demand from regulatory authorities, *compliance* has become a *must do activity* for every enterprise. Essentially, compliance corresponds to the enterprise's obedience to governing regulations enforced on its business operations. The demand for compliance can come from government regulations (e.g. the Sarbanes-Oxley Act, HIPPA, BASEL-III . . .), standards (ISO-9000, CoBIT . . .), and/or an enterprise's internal policies. Adherence with regulatory laws and internal controls essentially increase transparency and effective control over business operations.

Service-Oriented Architecture (SOA) is one of the enablers for innovation in today's highly competitive business environment. Public and private enterprises alike are adopting new technologies to bring innovations into their business operations and to offer their core competencies as web services. Web services are often physically independent but

* NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

logically interrelated pieces of services orchestrated to provide a specific functionality, and are designed by combining (possibly) disparate and often incongruous business processes from different enterprises [4]. In such a dynamic setting, the ability to trust that one another's internal processes that form the backbone of successful invocation of web services are compliant with regulations becomes even more crucial.

Business process models provide a high-level view on how business operations can be carried out to achieve a desired outcome. Business processes must behave within the defined limits of the regulatory guidelines (in legal context) called *norms*. Norms regulate business processes by imposing restrictions on *how business activities should be performed*. Any divergent behaviour may lead to termination of interactions or financial penalties [9]. Consider for example, a procurement process of a government agency which handles dynamic selection of vendors to place orders, which is implemented as a web service. Using such a web service, the agency can quickly place an order, receive and evaluate the quotes from suppliers. The procurement process is subject to regulations, as such the procurement web service must be checked for compliance with relevant regulations before it can be deployed. A process model that reflects the behaviour of the procurement web service can be used to verify the effectiveness of regulations and policy controls.

The structure and properties of norms have been extensively studied by the field of Deontic Logic, Artificial Intelligence and Law, and Legal Reasoning (see, [15] for a comprehensive treatment with a formal and legal theory perspective). A number of researchers have incorporated the notion of process compliance in the service domain. [13] deals with business rules driven business processes as service composition using various types of composition elements. The business rules considered in the framework are related to the structure of business processes. [18] provides a formal characterisation of behavioural rules for business policy compliance for SOA which is again useful to check structural compliance of business processes. But compliance is not only about how the activities should be performed (the control flow aspect) but about what these activities do (data), and who performs the tasks (resources aspect).

Generally the compliance rules are written in a natural language (c.f. those that can be found in legal documents or policy documents). To enable automatic compliance checks of processes, these rules need to be formalised in a machine-readable format. Typically the formalisation of compliance rules is language dependent, and the choice of a formal language depends on the business analysts. In this paper, we carefully examine all different types of normative requirements which can be imposed upon the different perspectives of business processes and propose how these requirements can be captured in a formal manner without restricting ourselves to any particular formalism.

Hence, the aim of this paper is not to provide yet another framework for business process compliance; instead we provide conceptually sound foundations for the normative requirements for the normative component of the compliance problem. This is achieved by giving semantics of norms (obligations) in terms of the validity of a norm, effects of the violations; and the possible ways in which a business process can be executed.

In the next section, we provide a motivating scenario of a complaints handling process together with a set of normative requirements. The formal definitions of business process models are given in Section 3. Various types of normative requirements together with concrete examples for each type are discussed in Section 4. An illustration of how compliance checking can be carried out for the complaints handling process as well as an evaluation of a compliance framework, Regorous, based on the proposed set of normative requirements is provided in Section 5. Section 6 concludes the paper.

2 Motivating Scenario: A Complaints Handling Process

In this section, we provide a short description of the complaints handling process from the Land and Property Management Authority (LPMA) in New South Wales, Australia. This process is required to follow a number of compliance requirements based on an internal policy document[2].

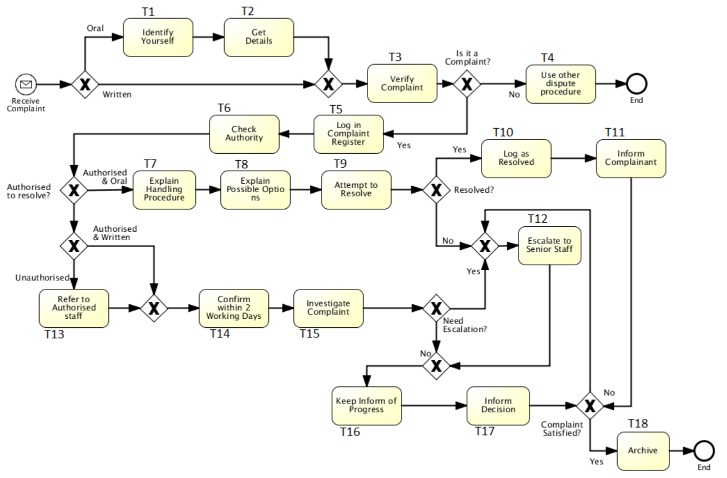

Fig. 1: Complaints Handling Process from LPMA, NSW Australia.

Figure 1 depicts the overview of the procedure followed to resolve a complaint as a BPMN process model. According to the guidelines the first step in the process is to determine whether a complaint is an oral complaint or a written complaint. If it is an oral complaint, a staff member will identify himself and details are gathered from the complainant before proceeding. The staff member then verifies whether the received complaint meets the requirements of a legitimate complaint as defined in Section 9 of the policy. If the received complaint does not meet the definition of a complaint, alternative dispute procedures are adopted (which is out of the scope of this process). After a complaint has been determined as a legitimate complaint, the staff member must decide whether (s)he has the appropriate authority to handle the complaint. If the staff is deemed to have the authority, then the complaint will go though the complaints handling process with the staff as its handler. Otherwise, the complaint is referred to an authorised staff and the complainant is informed. The authorised staff explains the process and the available options and attempts to resolve the complaint straight away if it is an oral complaint. If the complaint is resolved, then the complaint is logged as resolved and the complainant is informed about the decision.

[2] Available at: `http://www.lpma.nsw.gov.au/__data/assets/pdf_file/0004/25663/rth_Ch26_Aug_2009.pdf`

For a written complaint, an authorised staff will confirm the complaint within two working days. A complaint is escalated to a senior staff if it cannot be resolved or the complainant is not satisfied or if the staff decides that it needs to be escalated. While the complaint is being investigated, the complainant is being kept informed. When a decision has been reached, the complainant is informed about the decision. When the complainant is satisfied with the decision, the complaint is closed off and archived.

Table 1 shows the policy excerpt of the compliant handling process.

Table 1: The Compliance Requirements of Complaints Handling Process from LPMA, NSW.

Rule ID	**Policy Description (Compliance Control/Specification)**
R1	Staff receiving a complaint will aim to resolve it at the earliest opportunity.
R2	*Where the client is not satisfied with the initial response to the complaint, they will be given the option to progress the issue(s) through the formal complaints handling process outlined in department's complaints handling procedure.*
R3	*Staff will treat all complaints fairly and impartially, as is their obligation under the code of conduct.*
R4	*All complaints will be acknowledged within 2 working days of being initiated.*
R5	*All complainants kept informed about the progress of the matter, particularly if delays occur.*
R6	*Complainants will not be subject to any form of prejudice, loss of services, or be disadvantaged in any way as a result of having complained.*
R7	*Complaints will be treated with an appropriate level of confidentiality. Information about complaints will only be shared on a need–to–know basis, both within the agency and externally.*
R8	*Reasons will be provided for decisions made in relation to complaints received.*
R9	*If complaints do not meet the conditions in Section 9, the department may set limits or conditions on the handling of their complaints.*
R10	*Unauthorized staff cannot handle complaints (either oral or written).*

3 Formal Foundations of Business Process Compliance

Compliance is related to the behaviour of a process, where by the behaviour we understand how the process can be (*correctly*) executed. Thus we have to identify the traces of a process, where, from the compliance point of view a trace is the sequence of actions/tasks performed by the process. Compliance is not only about the tasks or actions undertaken but also what the tasks do, their artifacts and how they change the environment in which the process is situated. To capture this, we adopt the idea proposed in [14] and enrich processes with semantic annotations. These annotations are meant to capture the attributes, resources and other information related to the tasks in a process. We take an agnostic approach to the annotations themselves and assume that there is a suitable language to represent the annotations. We stipulate that the same language is used to represent both the annotations and the content of the normative requirements.

In this paper, we make use of *workflow-nets* (WF-Nets) [17], a subclass of Petri nets [12], to represent business processes. However, the definitions below can be easily modified for other representations of business processes.

Definition 1 (Petri Net). *A Petri Net is a tuple $PN = (P, T, F)$ where P is the set of places, T is the set of transitions, $P \cap T = \emptyset$ and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation.*

A Petri net is a collection of two types of nodes: *places* and *transitions*. Arcs connect one type of node to the other. For a node $x \in (P \cup T)$, $\bullet x$ denotes the set of inputs to x and $x\bullet$ denotes the set of outputs of x. The *state* of a Petri net is represented by a *marking* that describes the number of tokens in each place of a net.

A workflow net (WF-net) is defined as a subclass of Petri net with the following structural restrictions [16]. There is exactly one *source place* and exactly one *end place*. Every node in the graph is on a direct path from the source place to the end place.

Definition 2 (WF-net). *Given a Petri net $N = (P, T, F)$, the net N is a WF-net if and only if: (1) There is one source place $i \in P$ such that $\bullet i = \emptyset$. (2) There is one sink place $o \in P$ such that $o\bullet = \emptyset$. (3) Every node $x \in P \cup T$ is on a path from i to o.*

Definition 3 (Enabling and Firing Rules of a WF-net). *Given a WF-net $N = (P, T, F)$, a transition $t \in T$ and a marking M of N, t is enabled at M, denoted as $M[t\rangle$, if and only if, there is at least one token each in all $p \in \bullet t$. If $M[t\rangle$ holds and transition t is fired, a new marking M' of N is reached, which removes a token each from each $p \in \bullet t$ and puts a token in each $p \in t\bullet$. This is denoted as $M \xrightarrow{t} M'$.*

Definition 4 (Occurrence sequence). *Given a WF-net $N = (P, T, F)$ and markings $M, M_1, \ldots, M_n$ of N, if $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} M_n$ holds then $\sigma = \langle t_1, t_2, \ldots, t_n\rangle$ is an occurrence sequence leading from M to M_n.*

The initial marking of a WF-net is i, where there is one token in the source place i, and the end marking of a WF-net is o. A *trace* in a WF-net represents an occurrence sequence from the initial marking i to the end marking o.

Definition 5 (Labeled WF-Net). *A labeled WF-net $N = (P, T, F, l)$ is a WF-net (P,T,F) with a labeling function $l \in T \nrightarrow \mathcal{U}_A$, where $\mathcal{U}_A$ is some universe of activity labels. Let $\sigma_v = \langle a_1, a_2, \ldots, a_n\rangle \in \mathcal{U}_A{}^*$ be a sequence of activities and M, M' be two markings of N. $M[\sigma_v \triangleright M'$ if and only if there is a sequence $\sigma \in T^*$ such that $M[\sigma\rangle M'$ and $l(\sigma) = \sigma_v$.*

With this definition we only have the *visible and labeled* transitions in the net. For a set of traces of a workflow net $\mathfrak{T}^+(N)$, $\mathfrak{T}^+ = \{\sigma_\Theta | i[\sigma_\Theta\rangle o\}$ is the set of all visible traces in the net, where $\Theta = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a set of all occurrence sequences. The idea behind the notion of a labelled WF-Net is that a trace of visible transitions corresponds to a possible execution sequence of the process, where the visible transitions correspond to the tasks executed by the process.

Next, we look at how a WF-net can be annotated with compliance requirements. We begin with the definition of the language.

Definition 6 (Literal). *Let A be the set of all atomic propositions. The set of literals is $\mathfrak{L} = \{a, \neg a | a \in A\}$.*

A consistent set of literals can be understood as either a (partial) interpretation (i.e., an assignment of truth value) or equivalently a (partial) description of a state.

Definition 7 (Consistent Set). *A set of literals L is consistent if and only if L does not contain any pair of literals $l, \neg l$.*

Definition 8 (Annotation). *Let N be a WF-net and $\mathfrak{T}^+$ be the set of visible traces of N. An annotation ann is a function $Ann : \mathfrak{T}^+ \times \mathbb{N} \mapsto 2^{\mathfrak{L}}$ such that for every $t \in \mathfrak{T}^+$ and every $n \in \mathbb{N}$, $Ann(t,n)$ is a consistent set of literals.*

Annotations enable a process to have states attached to the tasks. The function $Ann(t,n)$ returns the state obtained after the execution of the n-th task (visible transition) in the (visible) trace t.

Definition 9 (Annotated WF-Net). *An annotated WF-net is a pair $\langle N, Ann \rangle$, where $N = (P,T,F,l)$ is a labeled WF-net, and Ann is an annotation.*

In an annotated WF-net, each visible trace uniquely determines the sequence of states obtained by executing that trace. Thus, in what follows whenever clear from the context we use trace to refer to a sequence of tasks, and the corresponding sequence of states.

Remark 1. It is not within the scope of this paper to describe how the sequences of states corresponding of the execution of a process are obtained. The task of specifying how the function *Ann* is implemented is left to specific compliance applications.

4 Normative Requirements

Norms regulate the behaviour of their subjects and 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. [5] provides a comprehensive list of normative effects. From the compliance perspective the normative effects of importance are the deontic effects. The basic deontic effects –from which others deontic effects can be derived, see [15]– are: *obligation*, *prohibition* and *permission*.

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 prohibition to the contrary does not hold.

Obligations and prohibitions are constraints that limit the behaviour of processes. The different between obligations and prohibitions and other types of constraints is that they can be violated. On the other hand, *permissions* are constraints that cannot be violated and thus, permissions do not play a direct role in compliance. Instead, 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

[3] Here we consider the definition of such concepts given by the OASIS LegalRuleML working group. `http://www.oasis-open.org/apps/org/workgroup/legalruleml/`.

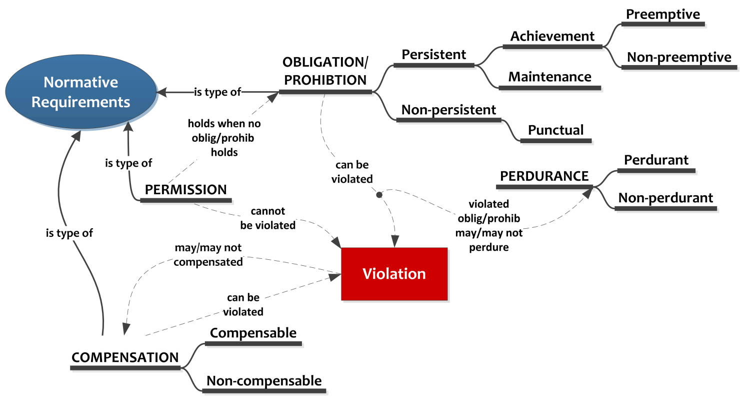

Fig. 2: Normative Requirements: Classes and Relationship

opposite of A), and then if A is obligatory, then $\neg A$ is forbidden [15]. 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 an *obligation*.

Compliance means to identify whether a process violates 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 norms give the conditions of applicability of obligations, the next question is *how long does an obligation hold for*. Essentially, a norm can specify that an obligation is in force at a particular time point only, or more often, a norm indicates when an obligation comes in force. An obligation is considered to remain in force until it is terminated or removed. In the first case we speak of *non-persistent obligations* and *persistent obligations* in the second.

A *persistent obligation* that needs to be obeyed for all time instances within the interval in which it is in force is a *maintenance obligation*. If achieving the content of the obligation at least once is enough to fulfill it, then it is considered an *achievement obligation*. For an *achievement obligation*, another aspect to consider is whether the obligation could be fulfilled even before the obligation is actually in force. If this is allowed, then we have a *preemptive obligation*, otherwise the obligation is a *non-preemptive obligation*. In contrast, a *non-persistent obligation* needs to be obeyed for the instance it is in force, and categorised as a *punctual obligation*. For *punctual obligations* the obligation contents are immediately achieved otherwise a violation is triggered.

An obligation of any type can be violated. A *violation* does not always imply the consequent termination of or impossibility to continue a business process. Certain violations can be compensated for, and processes with compensated violations are still compliant [7, 10]. For example, contracts typically contain compensatory clauses specifying penalties and other sanctions triggered by breaches of contract clauses [6]. However, not all violations are compensable, and uncompensated violations mean that a process is not compliant. The effects of a violation on the obligation that has been violated also need to be considered. If the obligation persists after being violated, it is a *perdurant obligation*, if it does not, then we have a *non-perdurant obligation*.

Figure 2 illustrates possibilities and relationships for the deontic effects we discussed in this section. The classification provided has been obtained in a systematic and exhaustive way when one considers the aspect of validity of obligations (or prohibitions), and the effects of violations on them, namely: whether a violation can be compensated for, and whether an obligation persists after being violated.

4.1 Modeling Obligations

In this section we provide the formal definitions underpinning the notion of compliance. In particular we formally define the different types of obligations depicted in Figure 2.

Definition 10 (Obligation in force). *Given a WF-net N, let $\mathfrak{T}^+$ be the set of visible traces of N. We define a function Force*: $\mathfrak{T}^+ \times \mathbb{N} \mapsto 2^{\mathfrak{L}}$.

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. Next, we define how and when the process has to fulfill the various obligations (depending on their type) to be deemed compliant.

Remark 2. As in Remark 1 we abstract from mechanisms to establish which obligations are in force and when. This is left for specific compliance implementations.

Definition 11 (Punctual Obligation). *Given a WF-net N and a visible trace $t \in \mathfrak{T}^+(N)$,an obligation o is a* punctual obligation *in t if and only if*

$$\exists n \in \mathbb{N}\colon o \notin Force(t,n-1),\ o \notin Force(t,n+1),\ o \in Force(t,n).$$

A punctual obligation o is violated in t if and only if $o \notin Ann(t,n)$.

A punctual obligation is an obligation in force in one task of a trace. The obligation is violated if what the obligation prescribes is not achieved in or done by the task, meaning that the literal not being in the set of literals associated to the task in the trace.

Definition 12 (Achievement Obligation). *Given a WF-net N and a visible trace $t \in \mathfrak{T}^+(N)$, an obligation o is an* achievement obligation *in t if and only if*

$$\exists n < m \in \mathbb{N}\colon o \notin Force(t,n-1),\ o \notin Force(t,m+1), \forall k\colon n \leq k \leq m, o \in Force(t,k)$$

An achievement obligation o is violated in t if and only if
- *o is preemptive and $\forall k\colon k \leq m,\ o \notin Ann(t,k)$;*
- *o is non-preemptive and $\forall k\colon n \leq k \leq m,\ o \notin Ann(t,k)$.*

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. For a preemptive obligation o we have a violation if no state before the last task in which o is in force has o in its annotations; while for a non-preemptive obligation the set of states is restricted to those defined by the interval in which the obligation is in force.

Example 1. 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. 8.2.1.a.i.B on the other hand is an achievement obligation since the clause specifies a deadline to achieve it. It is also a non-preemptive obligation as it is not possible to acknowledge a complaint before receipt. Clause (3) in Example 2 illustrates a preemptive obligation.

Example 2. 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 13 (Maintenance Obligation). *Given a WF-Net N and a visible trace $t \in \mathfrak{T}^+(N)$, an obligation o is a* maintenance obligation *in t if and only if*

$$\exists n < m \in \mathbb{N}\colon o \notin Force(t, n-1),\ o \notin Force(t, m+1), \forall k\colon n \leq k \leq m, o \in Force(t,k)$$

A maintenance obligation o is violated in t if and only if

$$\exists k\colon n \leq k \leq m, o \notin Ann(t,k).$$

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 a violation is triggered.

Example 3. 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, 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 (Definitions 14, 15, Definition 16) capture the notion of compensation of a violation. A compensation is a set of penalties or sanctions imposed on the violator, and fulfilling them makes amends 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. 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 14 (Compensation). *A* compensation *is a function* $Comp\colon \mathfrak{L} \mapsto 2^{\mathfrak{L}}$.

Definition 15 (Compensable Obligation). *Given a WF-Net N and a visible trace $t \in \mathfrak{T}^{+}(N)$, an obligation o is* compensable *in T if and only if $Comp(o) \neq \emptyset$ and $\forall o' \in Comp(o), \exists n \in \mathbb{N}\colon o' \in Force(t,n)$.*

Definition 16 (Compensated Obligation). *Given a WF-Net N and a visible trace $t \in \mathfrak{T}^{+}(N)$, 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 4 and 5 below illustrate, respectively, these two usages.

Example 4. 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 5. YAWL Deed of Assignment, Clause 5.2.[4]
Each Contributor indemnifies and will defend the Foundations against any claim, liability, loss, damages, cost and expenses suffered or incurred by the Foundations as a result of any breach of the warranties given by the Contributor under **clause 5.1**.

The final definition is that of a 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. If an 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.

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

Definition 17 (Perdurant Obligation). *Given a WF-net N and a visible trace $t \in \mathfrak{T}^+(N)$, an obligation o is a* perdurant obligation *in t if and only if*

$$\exists n < m \in \mathbb{N}\colon o \notin Force(t, n-1),\ o \notin Force(t, m+1), \forall k\colon n \leq k \leq m, o \in Force(t, k)$$

A perdurant obligation o is violated in t if and only if

$$\exists k\colon n < k < m,\ \forall j \leq k, o \notin Ann(t, j)$$

Consider again Example 1. Clauses TCPC 8.2.1.a.i.A and 8.2.1.a.i.B state 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.

4.2 Business Process Compliance

The set of (visible) traces of a given business process describes the behaviour 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 (visible) traces. Intuitively a process is compliant with a given set of norms if it does not violate the norms. As it is possible to perform a business process in many different ways, we can have two notions of compliance, namely:

> A process is (fully) compliant with a normative system if it is impossible to violate the norms while executing the process.
>
> A process is (partially) compliant with a normative system if it is possible to execute the process without violating the norms.

We have a fully compliant process if no matter in which way the process is executed, its execution does not violates the normative system. A partially compliant process is one where there is an execution of the process that does not violate the norms. Based on this intuition, we provide the definitions for trace compliance and process compliance.

Definition 18 (Compliant Trace). *Given a WF-net N and a trace t in $\mathfrak{T}^+$. Let $O(t)$ be the set of obligations in force in t, i.e., $O(t) = \bigcup_{n \in \mathbb{N}} Force(t, n)$.*

1. *A trace t is* strongly compliant *if and only if no obligation $o \in O(t)$ is violated in t.*
2. *A trace t is* weakly compliant *if and only if every violated obligation $o \in O(t)$ is compensated in t.*

Definition 19 (Compliant Process). *Let N be a WF-net.*

1. *N is fully compliant if and only if every trace $t \in \mathfrak{T}^+(N)$ is compliant.*
2. *N is partially compliant if and only if there exists a compliant trace $t \in \mathfrak{T}^+(N)$.*

Notice that a possible refinement of Definition 19 is possible to distinguish between strongly and weakly compliant processes. This is achieved by passing the strongly/weakly parameter to the traces. For example a process is strongly compliant if all its visible traces are strongly compliant.

The definitions given in this section (apart from the Definition 19) can be used across the entire life-cycle of a process: design-time, run-time and log analysis. As we pointed out in Remarks 1 and 2 the states and obligations in force have to be determined by specific compliance checking implementations. For example, the annotations associated to a task at run-time or log-analysis will be obtained from the running instance or extracted from the log and the data sources related to the process, while at design-time such information can be provided by business analysts or obtained from the schemas of the databases and data sources linked to the process. Definition 19 can be used at design time in what is called compliance-by-design [14, 10], i.e., verifying before deploying a process that the process complies with given regulations. Clearly, the definition is not suitable for checking compliance at run-time (also called conformance) or auditing (log analysis), since it is possible that some of the possible visible traces are never executed (run-time) or were not executed (auditing). For these two cases, one has to use Definition 18 instead applied to the executed traces, and to the traces of instances of a process recorded in a log.

5 Compliance Checking of the Complaints Handling Process

We now provide a concrete example of compliance checking based on the complaints handling process shown earlier. Table 2 describes the applicable compliance rules and their types. These rules are relevant to one or more tasks in the complaints handling pro-

Table 2: The RuleID and Types of Norms from the Complaints Handling Process.

Rule ID	Rule Type
R1	Obligation, Achievement, Non-preemptive, Non-perdurant
R2	Obligation, Achievement, Preemptive, Perdurant
R3	Obligation, Maintenance, Perdurant
R4	Obligation, Achievement, Non-preemptive, Perdurant
R5	Obligation, Achievement, Non-preemptive, Non-perdurant
R6	Obligation, Maintenance, Perdurant
R7	Obligation, Maintenance, Perdurant
R8	Obligation, Achievement, Non-preemptive, Perdurant
R9	Permission
R10	Prohibition, Maintenance, Perdurant

cess. For example, $Rule4$ is relevant to Task T_{14}, suggesting that all received complaints must be acknowledged within 2 working days when received. Similarly, $Rule9$ intends to verify the legitimacy of complaints which relates to Task T_3 in the process. Consider the following trace t.

$$t\colon \langle T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{18} \rangle$$

The obligation expressed by $R1$ is in force from Task T_5, and it will be associated with any following task until the obligation has been fulfilled. Whether $R2$ is relevant or not for a trace depends on the decision node after T_9, and it is not triggered in the trace given. Whereas $R3$ is in force from the beginning to the end of the process, and it is in all traces.

Evaluation

To conclude this section we report on an evaluation of the framework. Regorous is an implementation of the compliance checking methodology proposed by Governatori and Sadiq [14, 10] where the normative provisions relevant to a process are encoded in PCL [8, 9] and the tasks of a process are annotated with sets of literals taken from the language used to model the norms. The Regorous module to check compliance generates the traces of the given process and cumulates the annotations attached to tasks using an update semantics to determine the state corresponding to a task in a trace (i.e., in case a literal from the then current task is the complementary of from a previous task, we remove the old literal and we insert the new one). PCL offers support for all types of obligations described in the previous section, and for every steps in a trace, it retrieves the state corresponding to the task being examined. Based on state PCL determines the obligations in force for current task. Finally, it checks if the obligations have been fulfilled or violated based on the semantics discussed in the previous section. For the full details of PCL mechanisms, see [9].

Regorous was tested against the novel Australian Telecommunication Consumers Protection Code 2012. The code specifically mandates that every australian entity operating in the telecommunication sector has to provide a certification that their day to day operations complies with the code.

The test was limited to TCPC Section 8 concerning the management and handling of consumer complaints. The section 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 (literals). The formalisation of Section 8 required all types of obligations described in Section 4. Table 3 reports the number of distinct occurrences and, in parenthesis, the total number of instances (some effects can have different conditions under which they are effective).

Table 3: Number and types of obligations and permissions in Section 8 of TCPC

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)

The evaluation was carried over in cooperation with an industry partner operating in the sector of the code. The PCL formalisation of TCPC Section 8 was reviewed and informally approved for the purpose of the exercise by the regulator. 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 to capture the existing complaint handling and management procedures and other related activities covered by TCPC Section 8. 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. In the simplest and most frequent cases the modification required were just to ensure that some type of information was recorded in the databases associated to the processes. Other cases needed to addition to simple activities (tasks) either after or before other tasks (e.g., make customer aware of documents detailing the escalation procedure after an unsatisfactory outcome of a non-escalated complaint). The above two types of non-compliance were detected by unfulfilled achievement obligations and they were the results of new requirements in the 2012 version of the code. Another case of non-compliance was related to ensuring that a particular activity does not happens in a part of the process. Finally, there were some cases where combination of the above issue were needed (the novel way to handle in person or by phone complaints) where totally new sub-processes were designed.

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).

6 Conclusions

In the SOA and cloud computing domains, a number of approaches have offered several classifications of business rules for compliance checking. [1] classifies compliance rules from various regulatory frameworks for cloud-based compliant workflows. Spanning over nine categories their classification comprises three main rules classes relevant to either the control-flow or the data flow of workflow models. These rules classes are then formalised into Petri nets for automated detection of non-compliant behaviour. [3] provides a taxonomy of high level pattern-based compliance constraints for business processes. The compliance patterns are divided into three distinct classes of patterns; namely *atomic, composite, and timed*. These patterns are then formalised using temporal logic for generating the formal expressions for checking the compliance of business processes before actual deployment. Primarily the classification of normative requirements provided in these frameworks is useful for structural compliance checking only. In addition, these studies do not address *how to model* and *reason about* the normative component of compliance.

Contrary to that, we have provided its formal semantics in terms of what constitutes a violation, and this analysis was done based on the idea of (possible) executions of a process. In addition, for each type of normative requirement we have provided concrete examples from clauses of statutory/legislative acts corresponding to the requirement. With formalised compliance rules, we can specify the different types of rules describing various deontic modalities e.g. obligations, permissions etc. As result, business processes can be annotated with rules for compliance checking purposes. This means that any system (either SOA based or other) for checking whether real life business processes are compliant with real life regulations have to handle such all normative requirements.

One possible use of the framework is to compare different systems, logics, and frameworks for business process compliance. We plan to carry out such investigations. A second use is to study the (formal) properties of the problem of checking whether a business process is compliant. A first step in this direction is [2] proving that whether a structured business process (without loops) complies with a set of achievement obligations is already an NP-complete problem. Compliance is conceived as a type of soundness property of process, and thus the result must be compared to checking the soundness of process, and for the same class of processes (e.g., structured without loops) this can be done in linear time (see, e.g., [11]). This opens another area where the framework can be applied, namely to identify computationally tractable subclasses of the business process compliance problem.

References

1. R. Accorsi, L. Lowis, and Y. Sato. Automated Certification for Compliant Cloud-based Business Processes. *Business & Information Systems Engineering*, 3(3):145–154, 2011.
2. S. Colombo Tosatto, G. Governatori, P. Kelsen, and L. van der Torre. Business Process Compliance is Hard. Technical report, NICTA, 2012.
3. A. Elgammal, O. Turetken, W.-J. Heuvel, and M. Papazoglou. Root-Cause Analysis of Design-Time Compliance Violations on the Basis of Property Patterns. In *Service-Oriented Computing*, volume 6470 of *LNCS*, pages 17–31. Springer, 2010.
4. A. Elgammal, O. Turetken, W.-J. Heuvel, and M. Papazoglou. On the Formal Specification of Regulatory Compliance: A Comparative Analysis. In *Service-Oriented Computing*, volume 6568 of *LNCS*, pages 27–38. Springer, 2011.
5. T. F. Gordon, G. Governatori, and A. Rotolo. Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain. In *Proceedings of RuleML'09*, 2009.
6. G. Governatori. Representing Business Contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.
7. G. Governatori and Z. Milosevic. Dealing with Contract Violations: Formalism and Domain Specific Language. In *EDOC 2005*, pages 46–57, 2005.
8. G. Governatori and A. Rotolo. A Conceptually Rich Model of Business Process Compliance. In *APCCM'10*, volume 110, pages 3–12, 2010.
9. G. Governatori and A. Rotolo. Norm Compliance in Business Process Modeling. In *RuleML 2010*, pages 194–209. Springer, 2010.
10. 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.
11. B. Kiepuszewski, A. H. M. t. Hofstede, and C. Bussler. On Structured Workflow Modeling. In *CAiSE'00*, pages 431–445, 2000.
12. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
13. B. Orriäns, J. Yang, and M. Papazoglou. A Framework for Business Rule Driven Service Composition. In B. Benatallah and M.-C. Shan, editors, *Technologies for E-Services*, volume 2819 of *LNCS*, pages 14–27. Springer, 2003.
14. S. Sadiq, G. Governatori, and K. Namiri. Modeling Control Objectives for Business Process Compliance. In *Proceedings of BPM'07*, pages 149–164. Springer, 2007.
15. G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.
16. W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.
17. W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *BPM'00*, 2000.
18. H. Weigand, W.-J. van den Heuvel, and M. Hiel. Business Policy Compliance in Service-Oriented Systems. *Information Systems*, 36(4):791 – 807, 2011.