Inf Syst Front (2016) 18:429–455
DOI 10.1007/s10796-015-9558-1

Normative requirements for regulatory compliance: An abstract formal framework

Mustafa Hashmi[1] · Guido Governatori[1] · Moe Thandar Wynn[2]

Published online: 24 May 2015

Abstract By definition, regulatory rules (in legal context called *norms*) intend to achieve specific behaviour from business processes, and might be relevant to the whole or part of a business process. They can impose conditions on different aspects of process models, e.g., control-flow, data and resources etc. Based on the rules sets, norms can be classified into various classes and sub-classes according to their effects. This paper presents an abstract framework consisting of a list of norms and a generic compliance checking approach on the idea of (possible) execution of processes. The proposed framework is independent of any existing formalism, and provides a conceptually rich and exhaustive ontology and semantics of norms needed for business process compliance checking. Apart from the other uses, the proposed framework can be used to compare different compliance management frameworks (CMFs).

Keywords Norms · Normative requirements · Norms compliance · Business process regulatory compliance · Compliance frameworks

✉ Mustafa Hashmi
mustafa.hashmi@nicta.com.au

Guido Governatori
guido.governatori@nicta.com.au

Moe Thandar Wynn
m.wynn@qut.edu.au

[1] NICTA Queensland, 70-72-Bowen St. Spring Hill, Brisbane, Australia

[2] Queensland University of Technology (QUT), 2-George St. Brisbane, Australia

1 Introduction

In today's highly regulated corporate environment compliance has become an unavoidable activity for every enterprise. Compliance, in its broader sense, can be understood as enterprise's ability to meeting all the governing regulations enforced on its business operations. The demand for reporting compliance pressurises enterprises to streamline their business operations in accordance with the governing regulations. This demand has become even stronger because of big corporate scandals e.g., Enron, American Insurance Group etc., which resulted in the emergence of regulatory acts[1,2] and quality standards[3]. These acts and standards place restrictions and provide guidelines for enterprises to streamline their processes, and impose severe financial and criminal penalties otherwise.

Enterprises, public and private alike, are adopting SOA (Service-Oriented Architecture) based technologies to bring innovations into their business operations; and to offer their core compentencies as Web Service (WS). Often physically independent, web services are collections of interrelated services orchestrated to provide a specific functionality; and web services are designed by combining (possibly) disparate and often incongruous business processes from different enterprises (Elgammal et al. 2011). In such a dynamic setting, the ability to trust that one another's internal processes that form the core of successful invocation of web services are compliant with regulations becomes even more crucial.

[1] The US government. Sarbaes-Oxley Act, Public Law 107-204, 116 Stat. 745, 2002.

[2] Banking Committee on Banking Supervision (SCBS), BASEL-II Accord, 2004.

[3] ISO-9000: www.iso.org/iso/iso_9000

Business processes provide a high-level view on how business operations can be performed to achieve a desired outcome. Hence it is particularly important that business processes operate within the defined boundaries of the regulations (in legal context) called *norms*. Aiming to control the behaviour of business processes, norms impose restrictions on *how activities should be carried out*, and impose penalties for any divergent behaviour. Consider, for example, a procurement process of a government agency which handles the dynamic selection of contractors 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. This is subject to certain regulations, as such the procurement web service must be verified whether it is compliant with the relevant regulations before it can be deployed. Hence, a process reflecting the behaviour of a web service can be used to verify the effectiveness of the regulations and policy controls.

Governatori and Sadiq (2009) define business process compliance as the relationships between the formal specifications of a business process and the formal specifications of a set of normative constraints, where a process is compliant if the specifications of the processes do not violated the constraints formalising the norms. Accordingly, we have to provide (1) a formal model for the representation of business processes, (2) a formal model for the representation of the norms and, eventually (3) a bridging mechanism between the two representations (if they are expressed in two different formalisms).

In the recent years several works addressed the issue of (regulatory) compliance in the context of of business process management, service computing and cloud computing domains (see, (Becker et al. 2012; Fellmann and Zasada 2014) for recent surveys of existing approaches). The general idea is to determine whether the constraints (i.e., norms) imposed by some regulatory framework (ranging from statutory acts, to regulations, to industry standards, to best practices and internal policies) are met by some systems.

Regardless how good and feasible these approaches may be, to the best of our knowledge, the majority of the approaches neglect the aspect of whether the method they propose offers a faithful representation of the norms and it is suitable to reason appropriately with the norm. A non faithful representation of and inappropriate reasoning with the norm can have significant impact on the effectiveness of an approach.

The aim of this paper is to offer a formal foundational framework to evaluate the ability of a compliance framework to representing the norms a system has to comply with.

The structure and properties of norms have been extensively studied in the fields of Law and Legal Reasoning, Artificial Intelligence, and Deontic Logic (see, Sartor (2005) for a comprehensive treatment with a formal and legal theory perspective). In this paper, we concentrate on an aspect that, so far, has been left implicit or was captured in a procedural way by research on business process compliance, namely the description of the meaning of norms, or more precisely, of the effects of norms, in terms of business processes.

Since norms prescribe the conditions under which they are applicable and the effects when they are applied, a norm may be applicable for a certain period of time i.e., the duration when a norm enters into force and until when it is terminated. Hence no matter which process aspects a norm may be applicable to, norms can be classified according to their temporal aspect of validity and obligations arising from the violations of other norms. In this paper, we study the normative component of business process compliance and proposed an abstract formal framework comprising a classification of norms and formal semantics. In the classification, we examine various types of norms that can be imposed on different aspects of business processes in terms of temporal validity of a norms; and effects of violations on norms. Also, we examine how these norms can be modeled in a formal way. Our intention is not to propose yet another compliance checking framework but provide a conceptually rich foundation for the norms for legal component of the compliance problem. For this purpose, we provide *formal semantics in terms of states determining the temporal validity, what constitutes a violation, effects of violations on other norms* a process driven SOA system may be subject to, and the possible ways in which a business process can be executed.

The contribution of the paper is threefold:

Classification Model: The first contribution is a classification model for normative requirements. The classification has been obtained in a systematic and exhaustive way and provides a rich ontology of the various obligations modalities.

Formal Semantics: The second contribution is a formal semantics for norms (obligations) in terms of validity of a norm, what constitutes a violation of a norm and effects of the violations. The provided semantics are modelled independent of any specific formalism; and provide the basis for compliance checking approach.

Conceptual Evaluations: The last contribution is a conceptual evaluation of business process compliance frameworks using the classification model for normative requirements presented in Section 3. In the conceptual evaluation, we examined whether or not the existing frameworks provide support for all types of normative requirements proposed in our classification model.

The rest of the paper is organised as follows: next we provide formal foundations of business processes and workflow-nets (hereafter WF-nets) followed by a discussion (Section 3) on various types of normative requirements together with concrete examples from real-life legal documents. Then a complaint handling process as case study (Section 4) is discussed after which an illustration of the approach on how the compliance checking of the business processes can be carried out together with an evaluation based on the set of normative requirements is given (Section 5). A conceptual evaluation (Section 6) of the selected compliance management frameworks evaluated based upon an evaluation criteria highlights the major shortcomings of existing frameworks, and some previous studies (Section 7) have been discussed. In the last Section we give closing remarks and some pointers for future work.

2 Formal foundations of business processes

As we have discussed in the previous section business process compliance requires a formal model of the relevant business processes and a formal model of the relevant norms. In this section, we provide formal definitions of processes annotated with compliance requirements. In Section 3 we complete the picture by providing a formal model of norms based on the notions to be defined in this section. This would provide both the model of norms and the bridge between the formalisation of processes and that of norms. The final aim is to show the evolution of system or the environment in which a system operates, and check that the resulting states (and intermediate states) are compatible with the norms. In this section we show how to start from the notion of business process model to describe the sequences of states corresponding to the execution of the process. In the next section we use sequences of states to provide the semantics of different classes of norms, and to provide the defintions of what it means to comply with a norm, and to violate a norm.

Compliance is related to the behaviour of a process, that is, whether it is possible to correctly execute a business process. Compliance is not only about the the actions (i.e., tasks) undertaken during the execution of a process but also about their artefacts, and how the actions change the environment in which a process is situated. To capture this, we adopt the idea proposed by Sadiq et al. (2007) and enrich processes by means of semantic annotations, where an annotation is a formula in a formal language encoding an effect of a task. Conversely business processes only provide an abstract view that how the activities are performed, what activities (tasks) do in the process remains unclear in particular what effects are produced by a task at a particular state (or between states) when executed. Since business processes can be modeled by means of transition systems using various modeling languages, we adopt Petri-net based workflow-nets as a transition system to generate different states (traces) and subsequently semantically annotate these states with relevant information to know the state of affairs of the process execution.

In this paper, we make use of *workflow-nets* (WF-nets) as defined by van der Aalst (2000), a subclass of Petri nets (Murata 1989), to represent a business process. Definitions 1–4 are necessary to formally define a WF-Net and its behaviour. For other representations of a business process one can directly start from Definition 5 and the rest of the definitions in this section can be easily modified for other representations of a business process.

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 (van der Aalst 1998): there is exactly one source place and exactly one end place and 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 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 $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} M_n$ 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 labelled WF-net $N = (P, T, F, l)$ is a WF-net (P,T,F) with some labelling function $l \in T \nrightarrow \mathscr{U}_A$, where $\mathscr{U}_A$ is some universe of activity labels. Let $\sigma_v = \langle a_1, a_2, \ldots, a_n \rangle \in \mathscr{U}_A^*$ 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 labelled* transitions in the net. For a set of traces of a WF-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. One may, however, argue that there might some other (invisible) traces that may still affect the compliance checking of a business process model. However, invisible traces may consist of tasks representing invisible actions. These invisible actions are used for routing purposes only and may not represent any task from a business point of view (Gambini et al. 2011; Wen et al. 2010). In contrast, we use visible traces because tasks in a trace represent some activity and may have significance from a business perspective. Also, some literals representing obligations might be associated to the tasks of a trace. Hence for compliance checking, we limit our attention to visible traces only.

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 $\mathfrak{L} = \{a, \neg a | a \in A\}$.

In the rest of the paper we concentrate on consistent set of literals, where 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 $l, \neg l$.

The next step is to enable a process to have states attached to the tasks depending on which trace they appear in.

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.

The idea of the above definition is that $Ann(t, n)$ 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 (N, Ann), where $N = (P, T, F, l)$ is a labelled WF-net, and Ann is an annotation function.

Next, we illustrate the concepts behind the definitions presented in this section with a small example. As stated earlier, a process can be represented using any process modelling language (e.g., Business Process Modelling Notation (BPMN), Event Process Chains (EPC) etc.). Such a process model can be transformed into a Petri net/WF-net by making use of translation rules as shown in Dijkman et al. 2008; Ouyang et al. (2006, 2009). Figure 1 shows a simple BPMN process (with AND/XOR splits and joins) and its corresponding WF-net. Now, consider the abstract BPMN model in Fig. 1 as an emergency evacuation process with compliance requirements. Let's assume that Task A is '*sound alarm*', task B is '*alert people*', task C is '*inform fire services*', task D is '*contain fire*' and task E is '*evacuate place*'. Assuming that semantic annotations are written in some language, we consider the annotations consisting of two propositions: p meaning 'the alarm has sounded' and q meaning 'a small fire to contain'. Four possible traces of this process are as follows:

$t_1 : \langle A, B, C, D, E \rangle$,

$t_2 : \langle A, C, B, D, E \rangle$,

$t_3 : \langle A, C, B, E \rangle$,

$t_4 : \langle A, B, C, E \rangle$.

After the execution of task A, we have the state 'alarm has sounded' which can be represented as

$$Ann(t_1, 1) = Ann(t_2, 1) = Ann(t_3, 1) = Ann(t_4, 1) = \{p\}$$

for all traces. After executing the next two tasks B and C, also common to all traces, it is possible to have different annotations for these traces. For example, in traces t_1 and t_2, we reach

$$Ann(t_1, 3) = Ann(t_2, 3) = \{p, q\}.$$

In contrast, we reach the following state for t_3 and t_4,

$$Ann(t_3, 2) = Ann(t_4, 3) = \{p, \neg q\}.$$

In t_1 and t_2, we check whether the fire is small enough that it can be contained (task D) before evacuating (task E); otherwise we directly evacuate (task E) in t_3 and t_4. It can

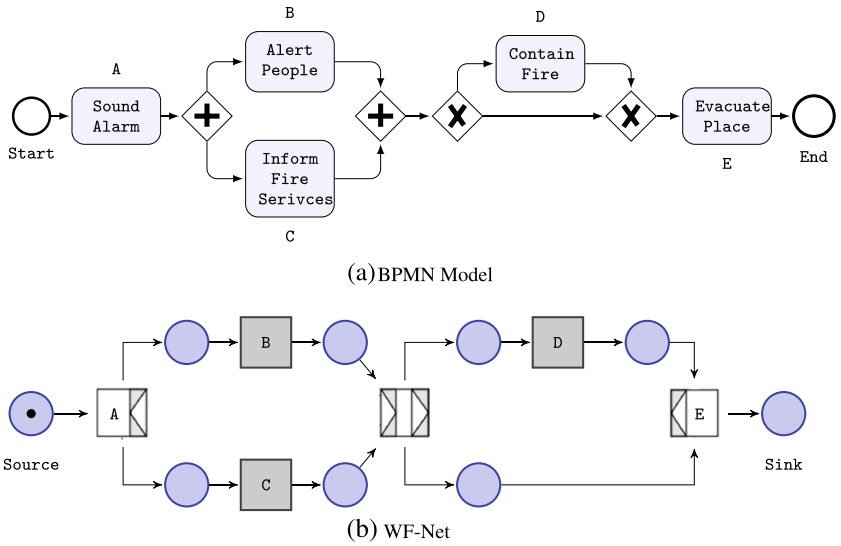

Fig. 1 Tranformation of the BPMN Model into an equivalent WF-Net

be seen that the information we have after the execution of tasks B and C varies depending on the trace being examined. For example from trace t_1 we know that the fire is small enough and it is possible to contain the fire represented as $Ann(t_1, 4) = \{p, q\}$. In contrast, trace t_3 informs us that it is not possible to contain the fire thus we have to evacuate, i.e., $Ann(t_3, 4) = \{p, \neg q\}$.

Note that different states can be obtained from different traces although the same tasks were being executed and the same end state can be reached from different traces. However, each visible trace uniquely determines the sequence of states obtained by executing the trace. Thus, in what follows whenever clear from the context, we use the term *trace* to refer to a sequence of tasks, and the corresponding sequence of states.

Remark 1 It is not 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 annotation function *Ann* is implemented is left to specific compliance applications. However, one can use the update semantics approach described in Ghose and Koliadis (2007) or by using the Event-Calculus (EC) to model the inertia of effects from a task to the next one as demonstrated in Goedertier and Vanthienen (2006) or by using the I-propagation approach for logical state representation described in Governatori et al. (2008); Hoffmann et al. (2012).

For example, if we take the process in Fig. 2 and Event-Calculus tasks can be represented by events, and propositions by fluents. Accordingly, the effect that p ('the alarm has sounded') holds after the execution of task A ('to sound alarm') can be represented by the domain specific axiom $initiates(p, A, T)$, meaning the event A initiates the fluent p at time T. The same correlation can be modelled by the formula $A \rightarrow \mathsf{X}\mathsf{G}p$, where X and G are the *next* and *always* operators of Linear Temporal Logic.

3 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. Gordon et al. (2009) provide a comprehensive list of normative effects. From a compliance perspective, the normative effects of importance are the *deontic effects*. The basic deontic effects are: *obligation*, *prohibition* and *permission*.[4]

Let us start by considering the basic definitions for such concepts:[5]

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

[4]There are other deontic effects, but these can be derived from the basic ones, see Sartor (2005).

[5]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.

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

Figure 2 illustrates the classification of the three basic deontic effects and the relationship between such effects and the notions of compensation and violation. The classification has been obtained by considering the validity of obligations (or prohibitions), and the effects of violations on them. Such considerations include whether a violation can be compensated for and whether an obligation persists after being violated. *Obligations* and *prohibitions* are constraints that limit the behaviour of processes. The difference between obligations and prohibitions and other types of constraints is that they can be violated. On the other hand, *permissions* are constraints that cannot result in a violation 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. Governatori (2015) gives a normative scenario where some deontic effects enter in force depending whether a permission is in force or not.

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 (Sartor 2005). 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 or not. 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 next question is how long does an obligation hold for. 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. Accordingly, in the first case we will speak of *non-persistent obligations* and *persistent obligations* in the second.

If a *persistent obligation* needs to be obeyed for the whole duration within the interval in which it is in force, it is categorised as a *maintenance obligation*. If achieving the content of the obligation at least once is enough to fulfil 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 Governatori and Sadiq (2009). For example, contracts typically contain compensatory clauses specifying penalties and other sanctions triggered by breaches of contract clauses (Governatori 2005). 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 considered a *perdurant obligation*, if it does not, then we have a *non-perdurant obligation*.

Next, formal definitions for these notions are provided together with examples taken from Acts and other legally binding documents.

3.1 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 depicted in Fig. 2 in relation to traces of business processes. In this way the definitions given below provide semantics of the normative requirements in terms of business processes.

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 fulfil to comply with a given normative framework. For example, $Force\ (t, 3) = \{p, q\}$ specifies that p and q are obligatory in the third task of trace t.

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

Remark 2 Similar 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.

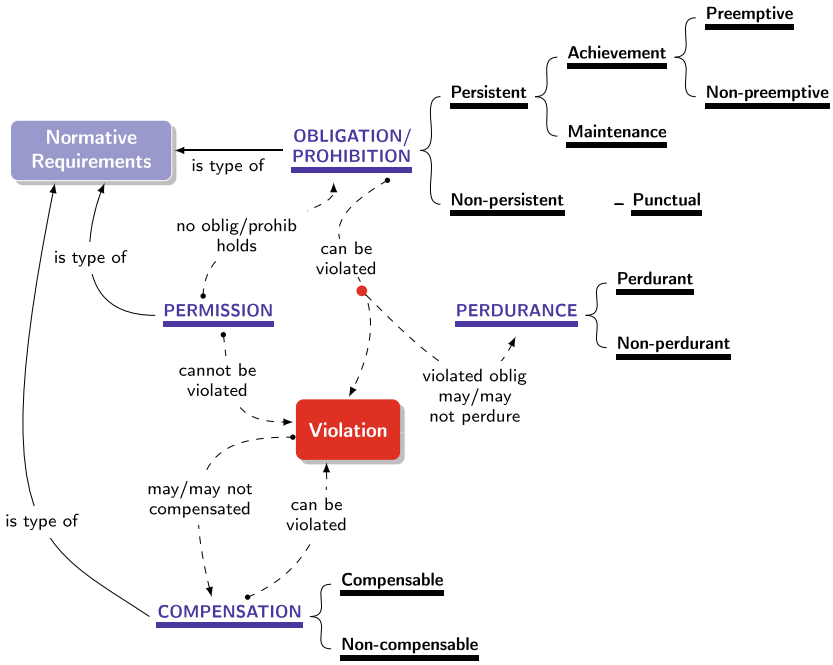

Fig. 2 Normative requirements: classes and relationship

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

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

The 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 $o \notin Ann(t, n)$.

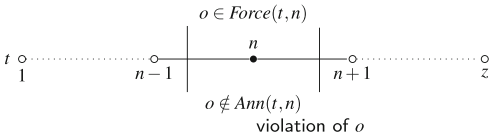

Diagram 3.1 Punctual Olbligation

Diagram 3.1 illustrates the nature of a punctual obligation.

A punctual obligation o (represented as a literal) is in force on one task n in a trace t, i.e., $o \in Force(t, n)$. Notice it might be the case that there are multiple instances in which the obligation is force. The obligation is violated if what the obligation prescribed is not achieved in or by the task when the obligation enters into force where this is represented by the literal not being in the set of literals associated to the task in the trace, i.e., $o \notin Ann(t, n)$ as shown in the diagram.

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

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

The obligation o is *in force between n and m*.

A persistent obligation is an obligation in force in an interval (a contiguous set) of tasks in a process. Diagram 3.2 depicts the definition where a persistent obligation o is in force between n and m at k-th task.

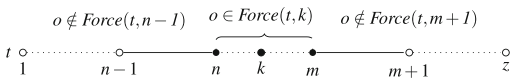

Diagram 3.2 Persistent Obligation

A persistent obligation can be further classified as *achievement* and *maintenance* obligations. The violation conditions for a persistent obligation can be derived from the violation conditions of these subclasses.

Definition 13 (**Achievement Obligation**) Given a WF-net N and a visible trace $t \in \mathfrak{T}^+$, 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

(a) o is *pre-emptive* and $\forall k : k \leq m,\ o \notin Ann(t, k)$;
(b) o is *non-preemptive* and $\forall k : n \leq k \leq m,\ o \notin Ann(t, k)$.

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.

Diagram 3.3 depicts the definition of a *preemptive obligation* in force at task k in a set of contiguous tasks between n and m, where m is the task when the obligation enters in force, and m is the deadline by when the obligation has to be discharged. The obligation o is violated if o does not in the annotations associated to all tasks preceding m. Notice that a preemptive obligation can be complied with even before the obligation is in force. Thus, one might ask why we bother with the task when the obligation enters in force. The reasons is that having (or not having) an obligation at a particular time could be the trigger of other deontic effects.

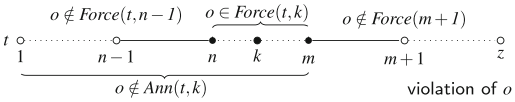

Diagram 3.3 Preemptive Obligation

For a *non-preemptive obligation*, the set of states one has to consider for determining whether the obligation has been violated is limited to those defined by the interval in which the obligation is force, see the pictorial representation of the non-preemptive case in Diagram 3.4.

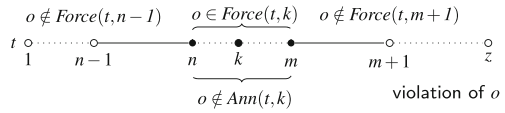

Diagram 3.4 Non-Preemptive Obligation

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 (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.

Next we give the examples illustrating the cases of *preemptive obligations*.

Example 2 Anti-Money Laundering and Counter-Terrorism Financing Act 2006. Clause 54 (Timing of reports about physical currency movements).

(1) A report under Section 53 must be given:

  (a) if the movement of the physical currency is to be effected by a person bringing the physical currency into Australia with the person—at the time worked out under subsection (2); or

[...]

  (d) *in any other case—at any time before the movement of the physical currency takes place.*

Clause (d) in this example describes that the *preemptive* obligation enters into force when a financial transanction happens, and the clause explicitly requires the report to be submitted to the relevant authority *before* the actual transaction (phyiscal movement of the currency) occurs. Notice that in some situations it might be the case that the transaction never occurres.

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.*

While clause (3) in this example prescribes *preemptive obligation* in the sense that it requires that a copy of the contract document is given to debtor but the obligation is fulfilled if the creditor provided a copy of the contract document earlier under clauses (2) of the section.

Definition 14 (**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 $\exists n, m \in \mathbb{N}, n < m$ 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

$\exists k : n \leq k \leq m, o \in Ann(t, k)$.

The following Diagram 3.5 illustrates the notion of a maintenance obligation.

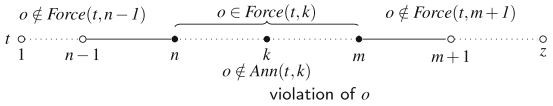

Diagram 3.5 Maintenance Obligation

Similar 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. Another difference is that deadlines are not required to detect the violation of maintenance obligations whereas for achievement obligation violations are detected at deadlines (Hashmi et al. 2014).

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. As in the above example, Telcos operators are prohibited to take credit management actions until a resolution of the complaint to the satisfaction of the customer is reached. The state where a credit management action does not occur must be maintained for all situations described by the norm until a resolution occurs.

The next three definitions are meant to capture the notion of compensation of a violation (see Diagram 3.6). The idea is that 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 in response to a violation of an obligation* (Definitions 15 and 16). 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 17).[6]

Definition 15 (**Compensation**) A *compensation* is a function $Comp: \mathfrak{L} \mapsto 2^{\mathfrak{L}}$.

Definition 16 (**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 $\forall o' \in Comp(o), \exists n \in \mathbb{N} : o' \in Force(t, n)$.

Definition 17 (**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.

Diagram 3.6 illustrates the notion of compensation. Assume that o is a compensable obligation in force in the interval delimited by n and m. Suppose the obligation is violated at d (say a deadline). The violation of the obligation triggers the entrance in force of the obligation(s) compensating the violation of o. In the diagram o' is one of compensations of o (since it belongs to $Comp(o)$), and it enters in force after d. Besides being a compensation of o' is just another obligation, thus it has the properties of the class of obligations it belongs to.

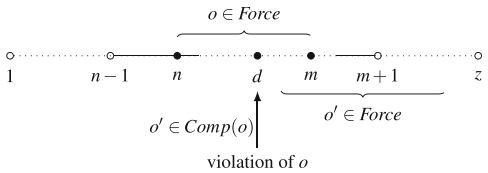

Diagram 3.6 Compensation Obligation

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 condition 2 from definition 17.

Compensations can be used for two purposes: to specify alternatives (i.e., less ideal outcomes) or to capture sanctions and penalties. Examples 5 and 6 illustrate these two usages respectively.

Example 5 TCPC 2012. Article 8.1.1.

A Supplier must take the following actions to enable this outcome:

[6]Notice that we took the most general definition by not imposing any temporal requirements for the compensation, thus the compensation could even precede the violation. Consider the natural language expression: "I apologise in advance for . . . ".

(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.[7]

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**.

Clause (B) of Example 5 and Example 6 respectively illustrate the case of compensation obligation. Clause (B) of Example 5 prescribes that a complaint should be escalated or managed under the Supplier's process if a complaint is not resolved as per the conditions of clause (A) of the section, allowing the customers to request for the escalation of their complaints thus compensates the violation of clause (A); in this case the compensation captures a behaviour the while not ideal it is still acceptable. While the obligation in Example 6 mandates the Contributes to compensate the Foundation for (negative) consequences incurred by the Foundations in case the conditions in clause 5.1 are violated. In this case the compensation imposes a sanction or a penalty.

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, 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 18 (**Perdurant Obligation**) Given a WF-net N and a visible trace $t \in \mathfrak{T}^+(N)$, an achievement obligation o is a *perdurant obligation* in t with a 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

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

[7] http://www.yawlfoundation.org/files/YAWLDeedOfAssignment-Template.pdf, retrieved on March 28, 2013.

Diagram 3.7 illustrates the notion of a *perdurant (preemptive achievment)* obligation. The obligation o is a perdurant with a deadline d is in force between n and m. o is not in $Ann(t, x)$ for all tasks x before d (d included). Thus we have a violation. Despite this the obligation remains in force till m. This means that the process still has to achieve o to be compliant (typically, in addition to compensations for the violation of the obligation).

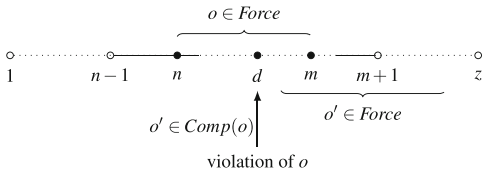

Diagram 3.7 Perdurant Obligation

Remark 3 Definition 18 only describes the perdurant obligation for preemptive achievement obligations. Simple adjustments can be made to model a similar notion for non-preemptive and maintenance obligations.

Consider again Example 1. 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 is violated, but the supplier still has the obligation to acknowledge the complaint. Thus the obligation in clause (i) is a perdurant obligation.

3.2 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. 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:

A process is (fully) compliant with a normative system if it is impossible to violate the norms 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 violates 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.

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

Based on the above intuition we can give the following definitions. We first define when a trace is compliant, and then extend that notion to cover a process.

Definition 19 (**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 20 (**Compliant Process**) Let N be a WF-net.

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

Notice that a refinement of Definition 20 is possible. Thus we can distinguish between strongly and weakly compliant processes. This is simply 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 (Definitions 10–20) given in this section (apart from Definition 20) 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 compliance applications and 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 (Hashmi et al. 2012).

Definition 20 can be used at design time in what is called *compliance-by-design* proposed by Sadiq et al. (2007); Governatori and Sadiq (2009), 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 19 instead applied to the executed traces, and to the traces of instances of a process recorded in a log.

4 Running example: a complaint handling process

In this section we give a short description of the scenario we are going to analyse in details in Section 5 to illustrate the how definitions given in Section 3 can be used to check whether a business process complies with a particular normative framework. The scenario is inspired by the internal complaint handling policy[8] from the Land and Property Management Authority (LPMA), New South Wales, Australia. In particular we describe a complaint handling process we designed to satisfy the a number of different types of compliance requirements obtained from the internal policy document, see Table 1. The first column shows the *rule ID*. The natural language description, the specific obligation type and deontic effects it may produce are given in the second column. Rule R_1, for example, is a *non-preemptive, non-perdurant achievement obligation* [OANPP], rule R_1 describes that any received complaint must be resolved at the earliest opportunity. Accordingly, the deontic effect (or obligation in force) for any received complaint for R_1 is the obligation *resolve_complaint*. Whereas R_4 specifies that all the received complaint must be acknowledged for which two options are provided: (1) immediately acknowledge complaint received in person or by a phone and (2) within two working days for a written complaint. Rule R_4 stipulates two different obligations i.e., a *punctual non-preemptive, perdurant obligation* [OPNPP] for (1) and a *non-preemptive, perdurant achievement obligation* [OANPP] for (2) respectively. The deontic effect R_4 produces is to acknowledge a received complaint, see Table 1 for the description, types and deontic effects of the rules related to the complaint handling process.

Figure 3 depicts the overview of the procedure followed to resolve a complaint as a BPMN process model. According to the policy guidelines, the first step in the process is to determine whether the received 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 with the complaints handling process. 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

[8]The policy document is available on the LPMA website: http://www.lpma.nsw.gov.au/__data/assets/pdf_file/0004/25663/rth_Ch26_Aug_2009.pdf

Table 1 The compliance requirements of complaints handling process from LPMA, NSW

Rule ID	Policy Description (Compliance Controls/Specifications)
R_1	Staff receiving a complaint will aim to resolve it at the earliest opportunity or at least confirm that complaint will receive attention. **Type:** Obligation, Achievement, Non-Preemptive, Non-Perdurant **Deontic Effect:** *resolve_complaint*
R_2	Where the client is not satisfied with the initial response to the complaint, they will be given the option to progress the issues through the formal complaints handling process outlined in the complaints handling procedure. **Type:** Obligation, Achievement, Preemptive, Perdurant **Deontic Effect:** *provide_escalation_options*
R_3	Staff will treat all complaints fairly and impartially, as is their obligation under the code of conduct. **Type:** Obligation, Maintenance, Perdurant **Deontic Effect:** *treat_fairly*
R_4	All complaints will be acknowledged: (1)immediately where complaints are made orally or by phone, (2)within 2 working days for written complaints. **Type-1:** Obligation, Punctual, Non-Preemptive, Perdurant **Type-2:** Obligation, Achievement, Non-Preemptive, Perdurant **Deontic Effect:** *acknowledge_complaint*
R_5	All complainants kept informed about the progress of the matter, particularly if delays occur. **Type:** Obligation, Achievement, Non-Preemptive, Non-Perdurant **Deontic Effect:** *inform_progress*
R_6	Complainants will not be subject to any form of prejudice, lose of services, or be disadvantaged in any way as a result of having complained. **Type:** Obligation, Maintenance, Perdurant **Deontic Effect:** $\neg$*disadvantage*
R_7	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. **Type:** Obligation, Maintenance, Perdurant **Deontic Effect:** *ensure_confidentiality*
R_8	Reasons will be provided for decisions made in relation to complaints received. **Type:** Obligation, Achievement, Non-Preemptive, Perdurant **Deontic Effect:** *provide_reasons*
R_9	If complaints do not meet the conditions in section 9, the department may set limits or conditions on the handling of their complaint. **Type:** Permission **Deontic Effect:** *limit_complaint*
R_{10}	Unauthorized staff cannot handle complaints (either oral or written). **Type:** Prohibition, Maintenance, Perdurant **Deontic Effect:** *authorized*

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 compliant 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.

For a written complaint, an authorised staff will confirm the process 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,

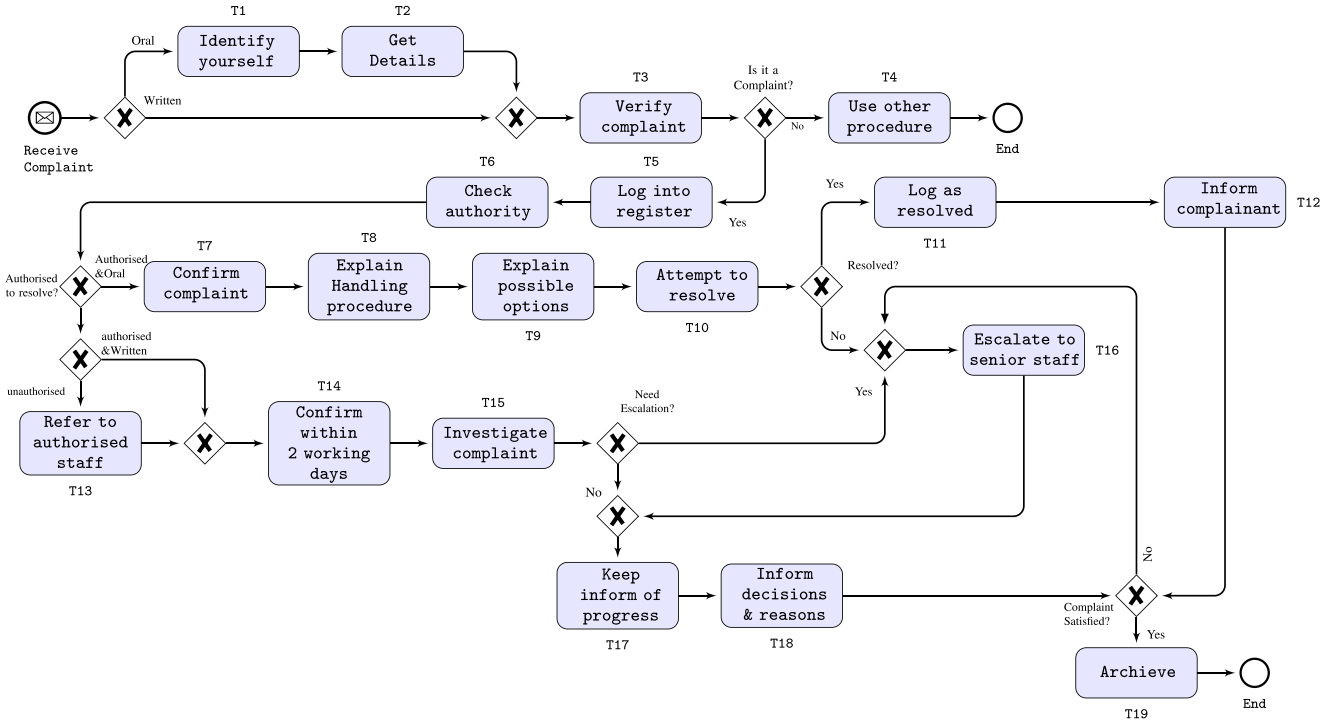

Fig. 3 A complaint handling process from LPMA, NSW Australia

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.

5 Compliance checking approach

Generally compliance rules are written in natural language (c.f. those that can be found in legal documents or policy documents). To enable automatic compliance checks, these rules need to be formalised in a machine-readable format. In addition, at the same time we need a machine-readable representation of processes and what the processes do in the various steps (and the states that would be generated by the processes when executed). Hence given a business processes and a set of norms (or compliance rules), checking whether a business process is compliant with the set of norms amounts to the following operations:

1. To determine the deontic effects (and their type) of the set of norms;
2. For each task in each trace of the process:
   (a) to determine what is the state corresponding to the task, and
   (b) to determine what are the obligations in force for the task;
   (c) to check whether the obligations in force have been fulfilled, violated (and for compensable obligation, whether they have been compensated for) or postponed the judgement to the next task in the trace, according to the semantics presented in Section 3.

This means that the problem of business process compliance reduces to that of populating the functions *Ann* and *Force*. As we discussed in the previous sections the aim of this paper is to provide a conceptually sound foundational framework for the semantics of regulatory complaints for business process compliance. The scope of this paper is not to propose a specific set of mechanisms, algorithms, or formalisms to check whether business processes are compliant, but the definitions given in this paper can be use to evaluate mechanisms, algorithms and formalisms proposed to check the compliance of business processes against relevant regulations.

Figure 4 illustrates the key concepts behind our approach for business process compliance. Rules impose conditions on the tasks to control the behaviour of processes. Business processes are annotated with rules for compliance checking purposes. These annotations are usually formalised rules and the data is parsed in the tasks at design-time.

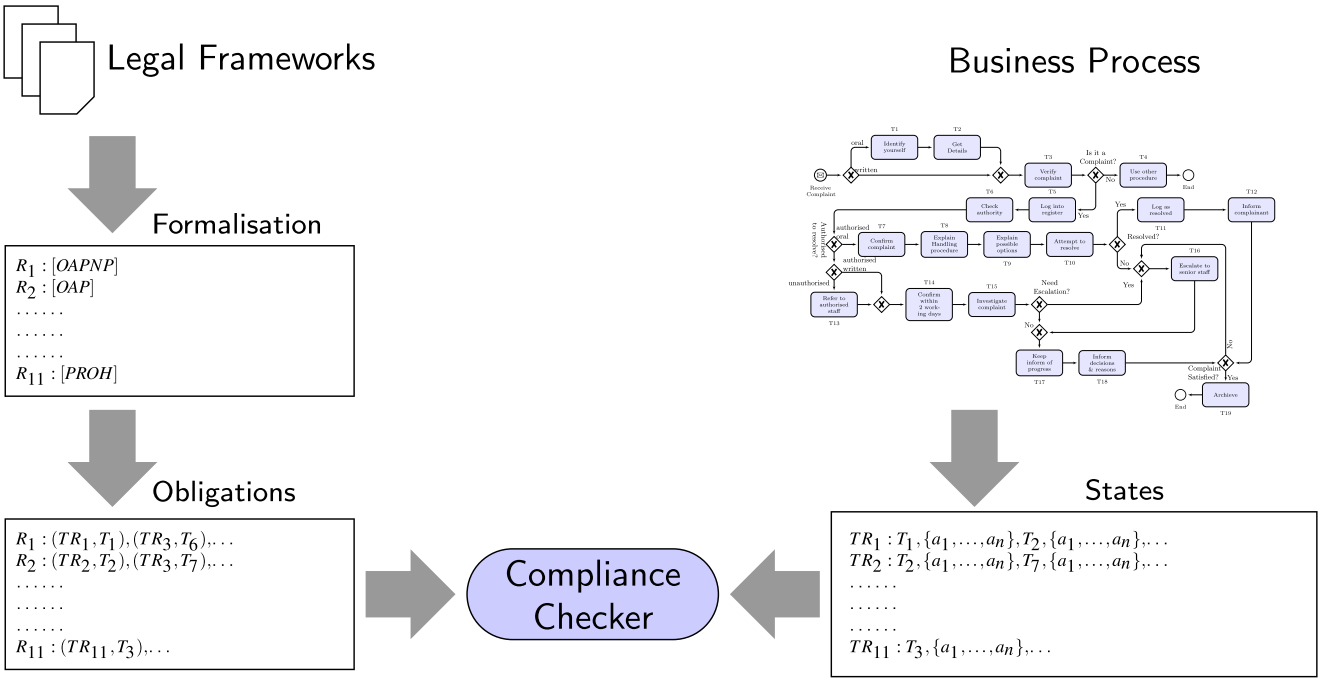

Fig. 4 Business process compliance: abstract framework

However at design-time, very limited information is available about the real data that a process operates on. Thus, for design-time compliance checking, business analysts provide abstract values and attributes to annotate processes. These abstract annotations can be used to verify the compliant behaviour of a business process at *design-time*. Contrary to that, at *run-time* processes are annotated with real values and attributes which are, again, provided by the business analysts. Regardless of checking the compliance at design-time or run-time, all we need is visible traces consisting of annotated tasks of a process.

In the rest of this section we are going to illustrate how the functions *Ann* and *Force* are populated in the scenario introduced in Section 4. Again, it not the scope of this paper to propose algorithms or formalisms to check whether business process complies with a set of compliance rules. The intention of the paper is to propose a language independent semantics to describe the properties of the compliance requirements in terms of business processes. It is up to different formalism to provide appropriate constructions to model compliance. For example, Article 8.2.1 of TCPC 2012 in Example 4, that prohibits a supplier to initiate credit management action when there is an unresolved dispute about credit, can be (roughly) represented in Linear Temporal Logic by the formula (where U is the Until operator)

$$\mathsf{G}(creditDispute \rightarrow (\neg creditManagementAction \mathsf{U} disputeResolved))$$

The until operator of Linear Temporal Logic is evaluated true if the left hand side of the operator is true for all instants before an instant where the right hand side of the operator is true. Thus the until operator respects the semantics for maintenance obligations

5.1 Compliance checking of the complaint handling process

We now provide a concrete example of compliance checking based on the complaint handling process shown earlier. Table 1 describes all the applicable rules on the complaint handling process. For each rule we have also identified the obligation triggered by the rule. These rules are of different types and relevant to one or more tasks in the aforementioned process. No compliance of every rule can be automatically checked because of several reasons e.g., the rule has been vaguely described or we have partial information only etc, see Awad (2010). Rule R_1 in the complaint handling process is one such type of rules that has been vaguely defined. For example, the *'earliest opportunity'* does not clearly specify until what time the obligation has to be fulfilled. However, rule R_1 is an achievement obligation applicable from T_3 and the obligation triggered by it remains in force until the obligation has been fulfilled. The rule R_4 is a punctual obligation (for oral complaint) and an achievement obligation (for written complaint) where the received complaint has to be acknowledged within 2 working days.

Rules R_3, R_6, R_7 are maintenance obligation applicable from the beginning of the process. They must be complied with for all the instances of the complaint handling process.

To determine whether the obligation has been complied with, regardless when an obligation comes into force and at which task in the process, one has to consider all the traces of the process including the task from where the obligation gets into force. Thus the first step is consider all the traces for a process. Given that there is a loop inthe process model, the number of traces is infinite. While this is not a problem for the theoretical compliance model, for practical purposes we have to consider a finite number of them. In practice loops typically have exit conditions, accordingly, we can limit the analysis to the case where each loop is expanded once, and in case of a nested loop, the external loop passes from the origin of the loop twice one where the internal loop is executed and the second when the internal loop is skipped. The this case the second time, the effect will be annotated with the exit condition. This procedure is applied recursively for more deeply nested loops.

We now list the elements of the (finite) set of traces $\mathfrak{T}_p$ generated by the compliant handling process.

$$
\begin{aligned}
t_1 &= \langle T_1, T_2, T_3, T_4\rangle,\\
t_2 &= \langle T_1, T_2, T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_3 &= \langle T_1, T_2, T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_4 &= \langle T_1, T_2, T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{17}, T_{18}, T_{19}\rangle,\\
t_5 &= \langle T_1, T_2, T_3, T_5, T_6, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_6 &= \langle T_1, T_2, T_3, T_5, T_6, T_{14}, T_{15}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_7 &= \langle T_1, T_2, T_3, T_5, T_6, T_{14}, T_{15}, T_{17}, T_{18}, T_{19}\rangle,\\
t_8 &= \langle T_1, T_2, T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_9 &= \langle T_1, T_2, T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{19}\rangle,\\
t_{10} &= \langle T_1, T_2, T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{11} &= \langle T_1, T_2, T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{16}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{12} &= \langle T_3, T_4\rangle,\\
t_{13} &= \langle T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{14} &= \langle T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{15} &= \langle T_3, T_5, T_6, T_{13}, T_{14}, T_{15}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{16} &= \langle T_3, T_5, T_6, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{17} &= \langle T_3, T_5, T_6, T_{14}, T_{15}, T_{16}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{18} &= \langle T_3, T_5, T_6, T_{14}, T_{15}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle,\\
t_{19} &= \langle T_3, T_5, T_6, T_{14}, T_{15}, T_{17}, T_{18}, T_{19}\rangle
\end{aligned}
$$

The next step is to determine what are the effects of the tasks in the trace as each task is annotated with one or more effects (or sets of effects), we refer to these effects as *annotations*. Hence we look which literals are relevant to each task in the trace. We use the *Ann* function defined in Section 2. To improve readability, in the rest of this section we use the annotation function *Ann* as:

$$Ann(trace, task, integer) = \{set\ of\ (consistent)\ literals\}$$

this means we also include the name of the task in its signature.

We now take a significative trace, trace t_{11}, to illustrate how the function populates the states corresponding to the tasks in a trace. Trace t_{11} is as follows:

$$t_{11} = \langle T_1, T_2, T_3, T_5, T_6, T_7, T_8, T_9, T_{10}, T_{16}, T_{17}, T_{18}, T_{16}, T_{17}, T_{18}, T_{19}\rangle$$

Based on the information available we show how the elements of trace t_{11} are populated[9]

$$
\begin{aligned}
&Ann(t_{11}, T_1, 1) = \{receive\_complaint, oral, identify\_yourself\}\\
&Ann(t_{11}, T_2, 2) = Ann(t_{11}, T_1, 1) \cup \{get\_details\},\\
&Ann(t_{11}, T_3, 3) = Ann(t_{11}, T_2, 2) \cup \{verify\_complaint\},\\
&Ann(t_{11}, T_5, 4) = Ann(t_{11}, T_3, 3) \cup \{valid\_complaint, register\_complaint\},\\
&Ann(t_{11}, T_6, 5) = Ann(t_{11}, T_5, 4) \cup \{check\_authority\},\\
&Ann(t_{11}, T_7, 6) = Ann(t_{11}, T_6, 5) \cup \{authorised, acknowledge\_complaint\},\\
&Ann(t_{11}, T_8, 7) = Ann(t_{11}, T_7, 6) \cup \{explain\_handling\_procedure\},\\
&Ann(t_{11}, T_9, 8) = Ann(t_{11}, T_8, 7) \cup \{explain\_options\},\\
&Ann(t_{11}, T_{10}, 9) = Ann(t_{11}, T_9, 8) \cup \{attempt\_resolution\},\\
&Ann(t_{11}, T_{16}, 10) = Ann(t_{11}, T_{10}, 9) \cup \{\neg resolve\_complaint, escalate\},\\
&Ann(t_{11}, T_{17}, 11) = (Ann(t_{11}, T_{16}, 10) - \{escalate\}) \cup \{inform\_progress\},\\
&Ann(t_{11}, T_{18}, 12) = Ann(t_{11}, T_{17}, 11) \cup \{inform\_decision, provide\_reasons\},\\
&Ann(t_{11}, T_{16}, 13) = (Ann(t_{11}, T_{18}, 12) - \{inform\_decision, provide\_reasons\})\\
&\qquad\qquad \cup \{\neg satisfied, escalate\},\\
&Ann(t_{11}, T_{17}, 14) = Ann(t_{11}, T_{16}, 13) \cup \{provide\_escalation\_options\},\\
&Ann(t_{11}, T_{18}, 15) = Ann(t_{11}, T_{17}, 14) \cup \{inform\_decision, provide\_reasons\},\\
&Ann(t_{11}, T_{19}, 16) = (Ann(t_{11}, T_{18}, 15) - \{\neg resolve\_complaint, \neg satisfied\})\\
&\qquad\qquad \cup \{satisfied, archive, resolve\_complaint\}
\end{aligned}
$$

The integer and task appearing in the *Ann* function indicate, respectively, the step of the process and the task (to be) executed at that step. Apart from its own, each task in the trace may inherit effects from its previous tasks to determine the state corres ponding to the task. These effects can be accumulated as the information grows for every subsequent task in the trace. These effects are computed based on the updated semantics where if the effects of previous tasks are in conflict with the effects of the current task, the effects of previous tasks are replaced with current ones. For example the state reached after task T_2, namely $Ann(t_{11}, T_2, 2)$, accumulates the effects of its previous task T_1 and also has its own effects $\{get\_details\}$. Similarly task T_3 accumulates the effects of tasks T_1 and T_2 producing $Ann(t_{11}, T_3, 3)$. In other cases, some effects, obtained in previous tasks can be remove or their truth value can be changed. For example

[9]The annotations for each task can be given by domain experts or can be extracted from databases or forms related to the tasks, see Hashmi et al. (2012).

Table 2 Applicable rules and obligations in force for trace t_{11}

Task, Step	Rules	Obligations in Force
$T_1, 1$	R_1, R_3, R_6, R_7	$Force(t_{11}, T_1, 1) = \{resolve\_complaint, treat\_farly, \neg disadvantage, ensure\_confidentiatly\}$
$T_2, 2$		$Force(t_{11}, T_2, 2) = Force(t_{11}, T_1, 1)$
$T_3, 3$		$Force(t_{11}, T_3, 3) = Force(t_{11}, T_2, 2)$
$T_5, 4$	R_4, R_5	$Force(t_{11}, T_5, 4) = Force(t_{11}, T_3, 3) \cup \{acknowledge\_complaint, inform\_progress\}$
$T_6, 5$	R_{10}	$Force(t_{11}, T_6, 5) = Force(t_{11}, T_5, 4) \cup \{authorized\}$
$T_7, 6$		$Force(t_{11}, T_7, 6) = Force(t_{11}, T_6, 5)$
$T_8, 7$		$Force(t_{11}, T_8, 7) = Force(t_{11}, T_7, 6)$
$T_9, 8$		$Force(t_{11}, T_9, 8) = Force(t_{11}, T_8, 7)$
$T_{10}, 9$		$Force(t_{11}, T_{10}, 9) = Force(t_{11}, T_9, 8)$
$T_{16}, 10$	R_8	$Force(t_{11}, T_{16}, 10) = Force(t_{11}, T_{10}, 9) \cup \{provide\_reasons\}$
$T_{17}, 11$		$Force(t_{11}, T_{17}, 11) = Force(t_{11}, T_{16}, 10)$
$T_{18}, 12$		$Force(t_{11}, T_{18}, 12) = Force(t_{11}, T_{17}, 11) - \{provide\_reasons\}$
$T_{16}, 13$	R_2, R_5, R_8	$Force(t_{11}, T_{16}, 13) = Force(t_{11}, T_{18}, 12) \cup \{provide\_escalation\_options, provide\_reasons\}$
$T_{17}, 14$		$Force(t_{11}, T_{17}, 14) = Force(t_{11}, T_{16}, 13)$
$T_{18}, 15$		$Force(t_{11}, T_{18}, 15) = Force(t_{11}, T_{17}, 14)$
$T_{19}, 16$		$Force(t_{11}, T_{19}, 16) = Force(t_{11}, T_{18}, 15)$

the first time we pass though task t_{17} (step 11) we remove the *escalate* flag which was raised in the previous task indicating the complaint was escalated. The change of polarity of literals is exemplified at step (16) where the negative $\neg$*resolve_complaint* and $\neg$*satisfied* are removed and replaced by their positive counterparts, *resolve_complaint* and *satisfied*.

Now we look at which rules are applicable to trace t_{11} on which task and when in order to determine which obligations are in force. Table 2 illustrates when the various rules become active in trace t_{11} (when they begin to produce their deontic effects), and when the various obligations are in force.

Four rules are effective at T_1. Rule R_1, whose deontic effect is an achievement obligation, becomes active as soon as a complaint is received, and remain active until the complaint is resolved. The other three rules, i.e., R_3, R_5 and R_6, are for maintenance obligations and never terminate for all instances of the process. No rules are associated with T_2, T_3, tasks T_7–T_{10} and with the tasks in the last three steps of the trace. Rules R_4 and R_5 produces achievement obligations, and their effects enter in force at step 4 (task T_5) when the complaint has been deemed valid. Rule R_{10} kicks in at task T_6, and its deontic effect is a maintenance obligation (that the staff is authorized to handle it, or alternatively the prohibition to handle a handle if not authorized).

Rule R_8 is triggered twice. The first time at step 10, and the corresponding non-preemptive obligation is in force for that step and the next one, when the obligation is fulfilled. Thus the obligation *provide_reasons* is no longer in force for step 12. The new decision in step 13, reinstate that non-preemptive obligation. The non-preemptiveness of the obligation implies that the previous discharging instance does not count for the instance of the obligation in force from step 13.

It is easy to verify that the trace is compliant for rules R_1, R_2, R_4, R_5, R_6, R_8, R_9 and R_{10}: The achievement obligations triggered by rules R_1, R_2, R_4, R_5, R_8 are fulfilled, respectively at steps: 16, 13, 5, 11, and 11 and 15 for the two instances of R_8. The maintenance obligation of rule R_8 is maintained from step 5, when it enters in force to the end of the process. R_9 is trivially complied with since

it is a permission, and it cannot result in a non-compliant situation. Finally, the maintenance obligations of rules R_3, R_6 and R_7 are not fulfilled. This is due to lack of information of what their obligations means in term of the given process.

5.2 Evaluation

To conclude this section we report on an evaluation of the framework against real processes and norms. The purpose of this section is to provide evidence that all types of obligations are eventually present in real life compliance scenarios.

The evaluation was carried out using Regorous.[10] Regorous is an implementation of the compliance checking methodology proposed by Sadiq et al. (2007): Governatori and Sadiq 2009 where the normative provisions relevant to a process are encoded in PCL Governatori and Rotolo (2010a, b) 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 the 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 Governatori and Rotolo (2010b).

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 distinct PCL (atomic) propositions (literals). The formalisation of Section 8 required all types of obligations described in Section 3. 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

[10] Regorous: Compliance Checker, available at https://www.regorous.com.

task long; in total there are over 1000 traces, and approximately 25000 states. 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 Conceptual evaluation of existing business process compliance frameworks

So far we have discussed the different types of obligations and provided the semantics for each obligation type in Section 3 and in Section 5 we have illustrated how these notions can be used to check whether a business process complies with a relevant set of normative requirements. We also discussed a number of existing CMFs (c.f. Section 1). In this section, we present the results of an evaluation conducted on seven CMFs using the classification model we proposed in Section 3. A summary of the evaluation results is shown in Table 4.

The evaluation was conducted to gain more understanding on the various aspects of a compliance framework especially from the legal knowledge (compliance requirements) representation perspective. Specifically, we were interested in knowing that how existing frameworks support normative requirements; what kinds of constructs are provided to model norms; which formalism is used, how normative requirements are linked to processes for compliance checking purposes; and what is the level of compliance management support in such frameworks etc.

Next, we first present our evaluation approach and then we discuss in details the frameworks we used to address these questions.

6.1 Evaluation approach

The aim of this section is to discuss the evaluation approach used to conduct the evaluations. We adopted a (systematic) *case study based shallow research approach*, which allowed us to start the evaluations 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 the frameworks based on the defined criteria. The details of each step are as follow:

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: To select a representative compliance management framework we defined a three steps framewok evaluation criteria which we used to conduct the evaluation as:

(a) *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,
(b) *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 obligation 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 the effects of violation on them,
(c) *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 El Kharbili 2012; Fellmann and Zasada 2014) rather we selected frameworks based on the 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.

6.2 PENELOPE

PENELOPE (*Process Entailment from Elicitation of Obligations and Permissions* Goedertier and Vanthienen (2006)) is a declarative language that captures obligation and permission constraints imposed on business processes by business policies. Aiming to provide design-time compliance verification capabilities, the language uses an algorithm that progressively generates the *state space* and *control-flow* of a business process. The state space generated contains a 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 the modelling of interactions between all involved partners and any from a third partner is represented as a time out event 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.

The deontic assignments in the PENELOPE are modelled 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. 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 business interactions.

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

6.3 Process Compliance Language (PCL)

PCL (Process Compliance Language) by Governatori and Rotolo (2010a) is a formal framework based on defeasible and deontic logic. It provides a conceptually rich formal foundations to model norms, and is able to efficiently capture the intuition of almost all types of normative requirements. These norms are modelled in the form of PCL rules for which the framework provides rich semantics. The state variables and the tasks in the process are represented by a set of propositional literals. PCL formulas, also called *PCL specifications* are written based on a set of primitive propositions using $\neg$ negation, $\otimes$ (a non-boolean connective modelling violations chains), and deontic operators representing obligations and permissions. The tasks in business processes are annotated with PCL specifications that are either provided by domain experts or are automatically extracted from the schemas of the databases or data sources linked to the processes using the technique proposed by Hashmi et al. (2012). These annotations are used to analyse whether the behaviour of an execution path is consistent with the annotated specifications. For this purpose, a three-step algorithm is used in which the process graph is first 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 behaviour. 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 all types of obligations as depicted in Table 4 and other aspects of normative reasoning. This is because the use of two types of logics where the 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 the defeasible logic (Governatori and Rotolo 2010b). To model the fundamental obligations, PCL provides three major constructs: punctual (O^p), maintenance (O^m), and achievement (O^a); achievement obligations are further refined in perdurant/non-perdurant and pre-emptive/non-preemptive.

Violations and obligations arising from the violations are major concerns in CMFs and 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.

6.4 DECLARE

Declare (DECLARE 2010) is a prominent framework for *run-time* verification of constraint-based declarative models. Declarative models describe *what a model can do* 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, Pesic and van der Aalst (2006)[11]), a language which provides graphical notations to model the flows of business interactions. Declare models (also called templates) are enacted by a workflow engine verifies the interactions among the tasks in the model to ensure compliance. 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 no violation of mandatory constraints and all constraints are fully satisfied at the end of the execution of an instance. The verification results of each constraint of an active instance are expressed as *satisfied*, *temporarily violated*, and *violated*. In case all constraints are satisfied the

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

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 modelled by means of Declare expressions which are classified as existence, relation, choice and negative constraints. The majority of these constraints are used to express obligations while the negative constraints can be used to express prohibitions. These constraints correspond to LTL expressions that provide the semantics to the Declare graphical notation. Currently, only achievement obligations and prohibitions can be modelled in the Declare Model, while no other norms types can be explicitly modelled. Since achievement obligations define deadlines and the obligation condition must be true 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, the different modalities of persistence and preemptiveness of obligations cannot be expressed. Expressing maintenance obligation constraints 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 deadlocks in the course of interactions among tasks. Declare is able to identify conflicts among constraints in the model, however it does not provide any support to handle violations because the expressions are written in LTL and the non-deterministic behaviour of the process models. Hence, in case of a violation the interaction among tasks in the Declare model will be stopped and no further activity can be performed. Accordingly, it is not possible to express permissions, compensations and perdurant obligations.

6.5 Business Process Modeling Notation-Query (BPMN-Q)

BPMN-Q (Awad et al. 2008, 2011) is a query based automated compliance checking framework capable of answering 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 a 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. Currently, the framework is able to handle almost the same types of obligations as Declare (cf. Table 4). While BPMN-Q is equipped with the so called *global space presence* pattern, this enables the execution of an activity that is required in all process instances, but this is not the same as an "activity" that must be performed for every single task in a process instance.

In BPMN-Q no conceptual or formal constructs for modelling permissions have been provided. Whereas prohibitions are represented by global space absence to prevent the execution of some activities. Unlike Declare, BPMN-Q is able to handle violations for which a violation handling approach has been discussed in Awad and Weske (2009). Finally, compensations and perdurant obligations are not supported because of using LTL as the underlying formalism to model compliance rules.

6.6 SEAFLOWS

SeaFlows (Ly et al. 2010, 2012) is a compliance framework that can be used for verification of semantics constraints. It incorporates a graphical language which provides 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 the compliance rules, the SeaFlows framework adopts a compositional graph-based modelling formalism allowing the modelling of the typical antecedent–consequence 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 the future by means of the occurrence pattern, the framework is not able to capture other types of deontic effects, e.g., punctual, maintenance, permission, and compensation (see Table 4. 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 (Herrestad 1991).

6.7 COMPAS

The COMPAS framework (COMPAS 2008) is a comprehensive compliance governance framework which provides an all-around compliance support for service-oriented-architecture (SOA) based systems. The framework adopts a model-driven development approach for designing compliant processes/services using a view based modelling framework and domain specific languages to model the compliance concerns in process models (Daniel et al. 2009). For compliance checking, business processes are annotated with compliance constraints in the form of (reusable) process fragments. These fragments underline the required behaviour of the control-flow of a process model, and are formalised using Linear Temporal Logic (LTL). Then the annotated process fragments are assessed to validate the compliant behaviour of the process models at run-time using event logs. A protocol component evaluates the generated event logs to check whether the process model complies with the behavior described in the attached compliance constraints process fragment. If the monitoring protocol detects any non-compliant behaviour it reports a violation and publishes it as a violation event. More recently some advances in the COMPAS framework (Elgammal et al. 2014) extended the modeling and reasoning support for the maintenance obligation, violation detection and handling. Also, the framework is now able to provide support for compensations obligations. Similar to Declare framework (cf. Section 6.4), COMPAS also use LTL as its underlying logic for modelling the normative requirements.

6.8 Business process compliance auditing framework

The business process compliance auditing framework (Ghose and Koliadis 2007) is a compliance checking framework to verify business process compliance against regulatory requirements. A local context description of accumulated effects is first defined by the analyst because the framework evaluates compliance locally at parts of the process where the effects are applicable. The effects accumulation process involves the derivation of a set of scenario labels at a point in the process (Hinge et al. 2009). Once the effects of relevant activities are accumulated and annotated to processes, the annotated processes are then encoded into directed graphs called semantic process networks (SPNs). These networks are used to verify properties related to the execution ordering of activities using an algorithm that exhaustively traverses all execution traces of the effect-annotated process model to check the rule violations. Then compliance results are reported (in boolean form) to indicate whether a process model satisfies the applicable compliance requirements or not.

The compliance requirements in this framework are annotated onto process models in the form of parsimonious effect-annotation. These effects annotations are parsed and modelled using a state-based logic, namely, Computational Tree Logic (CTL). Currently punctual, maintenance and compensation, permission and perdurant obligations are not modelled. It is not clear how the framework will model such

Table 4 Summary of normative requirements support in existing frameworks

	Obligations							
Framework	Punctual	Achievement	Maintenance	Compensation	Permissions	Perdurant	Prohibitions	Violations
PENELOPE	−	+	−	−	−	+	−	−
PCL	+	+	+	+	+	+	+	+
DECLARE	−	+	−	−	−	−	+	−
BPMN-Q	−	+	−	−	−	−	+	+
SEAFLOWS	−	+	−	−	−	−	+	+
COMPAS	−	+	+	+	−	−	+	+
AUDITING BPC	−	+	−	−	−	−	−	−

normative requirements for compliance checking because no conceptual and formal constructs have been provided.

6.9 Discussion

Table 4 shows the summary of the support provided by different compliance frameworks to model the different types of normative requirements. A '+' means that the framework provides direct support for the requirement, and a '–' means that the notion is not captured.

From the results presented in Table 4, it is clear that the vast majority of the evaluated CMFs is capable of supporting all types of norms and that only a fraction of normative requirements is widely supported. 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 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. However, its language is restricted to literals. DECLARE, BPMN-Q and COMPAS 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 and prohibitions only. Generally it is highly desirable that a formal language for compliance covers most of the properties and properties of the environment of the unit under verification (e.g., normative requirements). 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 conceptual results portray somewhat a *bleak picture* when it comes to see how existing frameworks represent legal knowledge for compliance checking because none is able to support all types of normative requirements. This does not, however, necessarily mean that the framework does not have the expressive power to model the notion, but the concept is not considered or analysed in that framework, including the case where the deontic effects cannot be faithfully represented. (Governatori 2015) provides an example where not paying attention to legal reasoning principles leads to results opposite to what legally trained professionals would conclude. This implies that adopting formalisms that are non conceptually grounded in legal practice creates framework that are unreliable and not suitable to be used in real-life applications.

7 Related work

We divide the related work section into two parts. We first discuss some papers that address the business process compliance problem and propose various techniques to solve the problem. We then describe some of the works which reported their results on the evaluations of existing compliance frameworks and position our work on the evaluation of compliance frameworks among them.

7.1 Process compliance in SOA/Cloud computing

In the SOA and cloud-computing domains there is considerable amount of approaches focusing on the compliance management of SOA based workflows. Also, these approaches offered several classifications of business rules for compliance checking purposes. Accorsi et al. (2011) classifies compliance rules from regulatory frameworks for cloud-based compliant workflows. Spanning over nine categories their classification comprises three main rule classes relevant to either the control-flow or data-flow of workflow models. These rules are then formalised in Petri nets for automated detection of non-compliant behaviour. Whereas Rodrìguez et al. (2013) proposed a SOA-enabled compliance management framework for auditing the compliance of SOA based processes. The authors used compliance templates enabling the detection of non-compliant behaviour of processes. These templates are annotated with the business constraints written in first-order-logic. Elgammal et al. (2010, 2012) provide 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. Orriëns et al. (2003) dealt with business rules driven business processes as service composition using various types of composition elements. The business rules considered in their framework are related to the structure of a business process. Weigand et al. (2011) provide a formal characterisation of behavioural rules for business policy compliance for SOA which is again useful for checking the structural compliance of business processes. While Ramezani et al. (2012, 2013) identified 54 control-flow based compliance rules distributed over 10 categories and 15 temporal rules distributed over 7 categories and proposed a compliance checking approach. These rules are mainly intended for compliance checking of business processes from structural and temporal aspects of a business

process only. In addition, these studies do not address how to model and reason about the normative component of compliance. In contrast, the classification we provide can be used both for structural and non-structural compliance of business processes. Moreover, we argue that first order logic is not suitable for modelling the legal norm as used by Rodrìguez et al. (2013).

7.2 Existing compliance approaches

We provide a quick overview of other existing compliance approaches not examined in Section 6.

Hee et al. (2010) propose a monitoring system for on-the-fly auditing of a business process. The proposed monitoring system is built either as labelled transition system with an infinite state space, or as a coloured petri net with tokens that grow unboundedly in size. The system operates in parallel to the business information system (BIS), and checks whether the essential business rules are complied with or interrupts the BIS to prevent the occurrence of a violation. From a set of business rules, an executable process model (i.e. a labelled petri net) is generated using a business rules language (BRL). The monitoring system tries to execute the events of the generated process model. If the monitor is able to execute the event the BIS operates without any interruptions, otherwise the violation of a business rule is reported and BIS is interrupted from processing the event any further. The proposed approach provides a good process monitoring mechanism for a deviant behaviour, the scope of this approach is for modelling as well as compliance checking support for all different types of normative requirements because it only supports a subset of business rules related to the authorisation and resources.

Ramezani et al. (2013) report a conformance checking approach based on Petri-Net patterns and alignments. The authors created a repository of 55 control-flow based compliance rules spanned over 15 distinct categories including compliance rules for data, resource and organisational rules. The collected rules were formalised in terms of Petri-nets rather than logics. For conformance checking, they employed alignment techniques from van der Aalst et al. (2012) to analyse if the process is compliant with the formalised Petri-Net patterns. If the patterns are consistent with the compliance rules, the execution behaviour is consistent. However, if any deviant behaviour is observed a violation of the rule is reported and the alignment shows the reason(s) for the deviations. The approach is promising for checking the compliance of control-flow related rules, but this only provides the structural compliance of the rules. In addition, conformance checking of business processes against the business rules have different specifications and properties from those of the legal domain. Thus the proposed approach is not suitable for compliance checking of the normative requirements.

The works by Elgammal et al. (2014); Mulo et al. (2013) provide more comprehensive compliance checking frameworks for business processes and incorporate the whole spectrum of business process aspects. Mulo et al. (2013) proposes a domain specific language (DSL) for event-based compliance monitoring for process driven SOA. The DSL implements the specifications of compliance directives imposed on the business processes. Whereas Elgammal et al. (2014) propose a compliance request language (CRL) grounded on linear temporal logic for design time compliance checking. Their language includes the series of compliance requirements patterns for all aspects of a business process. Although these approaches significantly improve the compliance checking for the normative requirements yet have are not able to fully cover all types of legal norms. For example, the CRL is still not able to model a punctual obligation and the obligations perduring after the violations whereas, with the DSL punctual, maintenance, compensation and obligations cannot be modeled.

Jiang et al. (2013, 2014) proposed a consistency and compliance-checking framework (CCCF) using the norms nets (NN) and colored petri nets (CPNs). The NN are used to formalise the regulatory rules and their relationship whereas the CPNs semantics implement the compliance checker toolbox. The CCCF framework provides the information whether a set of regulations is consistent and whether the business processes comply with the imposed regulations. Although the framework is able to provide a reasonable degree of automated support for verifying the compliance to regulation, however the transformation of the legal rules into NNs is interpreted primarily manually. In addition, from a business process perspective, the transformation of the model event sequences modeling the behaviour of the agent i.e., trace generation is also manual making the proposed framework less effective. Another downside of the framework is that there is no mechanism for modeling the temporal constraints in CPNs thus the compliance to regulation with temporal modality cannot be verified.

Letia and Groza (2013) reports a logic-based model checking approach for compliance verification of the integrated business processes models. The proposed approach extends the norm temporal logic of Ågotnes et al. (2007) and introduced obligations and permissions operators into the temporal logic to model the various compliance requirements from HACCP standard[12] in the food safety domain. The compliance checking is performed by a four steps

[12]The Hazard Analysis Critical Control Point System, available at http://www.standards.org/standards/listing/haccp, retrieved 20 Feb 2014

mechanism, where in the first step the domain knowledge i.e., the normative requirements, is translated into Norms Temporal Logic and Attribute Language with Complement (NTL-ALC) logic, then a WF-net using a Kripke structure is generated with states which are labeled with the all normative requirements are specified in the form of normative formula f pertaining to the state. The each formluma f in the state in the WF-net is verified if the formula f representing the norm holds in the state. If f does not hold the state violating the norm is added to the set of breached states. The proposed approach allows the integration of subsumption-based reasoning with the possibility to check the compliance of various types of norms. By the virtue of the extended logic NTL-ALC, the proposed approach allows the integration of abstract and the concrete business processes making it more explicit in representing the compliance requirements into business process models. This approach is very close to our approach presented in this paper, however it has limited scope in providing the modeling and reasoning support for all obligations classes discussed in Section 3.

Maggi et al. (2011) examine a declarative conformance checking framework which models the business knowledge defined in terms of constraints using a graphical language to model the flow of business interactions. The framework allows mandatory and optional constraints and a process instance is only active when there is no violation of the mandatory constraints and all the constraints are satisfied at the end of the instance execution. The business constraints in this framework are modelled by means of declarative expressions which are grouped as existence, relations, and choice and negative constraints. These constraints correspond to the LTL expressions which provide semantics to the graphical notation. The majority of these constraints are used to express obligations and prohibitions only. Whereas not all the classificatory classes of the our proposed classification model can be modeled using their LTL expressions.

7.3 Existing evaluation frameworks

Becker et al. (2012) offer a literature survey based on the generalisability and applicability of business process compliance frameworks. The evaluation is based on the reported implementation results from the surveyed frameworks, while El Kharbili (2012) 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, Cabannilas et al. (2010) study various frameworks using a four point criteria including the study of modelling languages that are used to model business processes and rules. Whereas Fellmann and Zasada (2014) surveyed 84 business process compliance approaches from their scope, phases of the process lifecycle and the trends of compliance research in variety of domains. The authors classified the existing compliance approaches using a four dimensions criteria. Elgammal et al. (2010) report a comparative analysis between formal languages to analysis how the compliance requirements are specified for automatic verification while (Turki and Bjekovic-Obradovic 2010) investigate the practice of regulation analysis and the approaches aiming to achieve and maintain regulatory compliance of given normative systems from an information systems and services perspective. Bonatti et al. (2004) study the existing approaches to logic and rule-based systems behaviour specifications from business and security policies rules to identify the possible usage for rule-based policies in a semantic web context. Addressing the issue of how to compare and evaluate compliance monitoring approaches, Ly et al. (2013) report on an evaluation of five frameworks from various domains using a set of core compliance management functionalities derived from the compliance literature and various case studies. Primarily their evaluation has a limited scope and lacks generalisability because it covers compliance monitoring frameworks only and excludes design-time compliance checking frameworks, which is a preferable compliance checking method these days.

Our evaluation is complementary to and different fromthese 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, using the classification of normative requirements we also examined whether or not existing CMFs can provide reasoning support for all types of normative requirements.

8 Conclusions and outlook

Legal norms have been studies from others fields e.g. Law and Legal reasoning but in the areas of Business Process Management and Service-Oriented Architecture (SOA) less attention has been paid. Since SOA enabled business process are subject to strict regulations for effective and transparent operations, in this paper we examine the various types of normative requirements for determining whether a business process complies with a normative documents (where normative document can be understood in a very broad sense, ranging from policies internal to an organisation, to best practice policies, to statutory acts). Primarily, most of the approaches have focused on compliance rules which are useful from a structural compliance of business processes. Also, not many studies address *how to model* and *reason about* the normative component of compliance.

Contrary to that, in the presented abstract framework, we studied the normative component of the business process compliance problem and provided a classification model of normative requirements, their formal semantics in terms of validity of a norm, what constitutes a violation, and the effects of the violations on the business processes. This analysis done in the framework was based on the idea of (possible) executions of a business process. In addition, for each type of the normative requirements we have provided concrete examples from clauses of statutory/legislative acts corresponding to the requirements. 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 the rules for compliance checking purposes. This means that any system (process-driven SOA or any other) for checking whether real life business processes are compliant with real life regulations have to handle all of such normative requirements. To validate the effectiveness of our proposed classification of normative requirements and compliance checking approach, we used a complaint handling process and other case examples; and practically demonstrated how the compliance of business processes can be checked annotated with the compliance rules.

The reported framework can be used in a number of ways: (1) it can be used to compare different systems, logics and frameworks for business process compliance. We have used the classification model to check the conceptual foundations of existing compliance frameworks, and plan to carry out further investigations, (2) it can be used to study the (formal) properties of the problem of checking whether a business process is compliant. A first step in this direction is the work by Colombo Tosatto et al. (2014) 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, Kiepuszewski et al. (2000)). This opens another area where the framework can be applied, namely to identify computationally tractable subclasses of the business process compliance problem. Since in presenting this framework we did not restrict ourselves to any particular formalism, the framework is generic in the sense that any formal language could fit in the framework despite the fact that we grounded it on deontic logic in mind. To validate this fact whether the classes of normative requirements and formal semantics presented in this work can be efficiently modeled with other formalisms we proposed a Event-Calculus based norms modeling framework in Hashmi et al. (2014). We intend to continue this work using other formal languages e.g., first-order-logic, temporal logic etc., for more comprehensive validation.

Acknowledgments This paper revises and extends ASSRI'13 (Hashmi et al. 2013) and AP-BPM 2013 (Hashmi and Governatori 2013) conference papers respectively. NICTA is funded by the Australian Government by the Department of Communication and the Australian Research Council through the ICT center of Excellence program.

References

Accorsi, R., Lowis, L., & Sato, Y. (2011). Automated Certification for Compliant Cloud-based Business Processes. *Business & Information Systems Engineering*, *3*(3), 145–154. doi:10.1007/s12599-011-0155-7.

Ågotnes, T., van der Hoek, W., Rodríguez-Aguilar, J.A., Sierra, C., & Wooldridge, M. (2007). On the logic of normative systems. In *Normative multi-agent systems, 18.03. - 23.03.2007*. http://drops.dagstuhl.de/opus/volltexte/2007/921.

Awad, A. (2010). A *compliance management framework for business process models*. PhD thesis, HPI, Potsdam University, Germany.

Awad, A., & Weske, M. (2009). Visualisation of compliance violations in business process models. In *5th workshop on business process intelligence* (Vol. 9, pp. 182–193).

Awad, A., Decker, G., & Weske, M. (2008). Efficient compliance checking using BPMN-Q and temporal logic. In *BPM, LNCS* (pp. 326–341). Springer.

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

Becker, J., Delfmann, P., Eggert, M., & Schwittay, S. (2012). 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.

Bonatti, P.A., Shahmehri, N., Duma, C., Olmedilla, D., Nejdl, W., Baldoni, M., Baroglio, C., Martelli, A., Coraggio, P., Antoniou, G., Peer, J., & Fuchs, N.E. (2004). *Rule-based policy specification: state of the art and future work*. REWERSE Project Report-i2-D1.

Cabannilas, C., Resinas, M., & Ruiz-Cortes, A. (2010). Hints on how to face business process compliance. In *III Taller de Procesos de Negocio e Ingenieria de Servicios PNIS10 in JISBD10* (Vol. 4, pp. 26–32).

Colombo Tosatto, S., Governatori, G., & Kelsen, P. (2014). Business process regulatory compliance is hard. *IEEE Transactions on Services Computing PP(99)*, 1–1. doi:10.1109/TSC.2014.2341236.

COMPAS (2008). Compliance driven models, languages, and architectures for services. In *7th framework programme for ICT*.

Daniel, F., Casati, F., D'Andrea, V., Mulo, E., Zdun, U., Dustdar, S., Strauch, S., Schumm, D., Leymann, F., Sebahi, S., de Marchi, F., & Hacid, M.S. (2009). Business compliance governance in service-oriented architectures. In *International conference on advanced information networking and applications, 2009. AINA '09* (pp. 113–120).

DECLARE (2010). *Declarative process models*. http://www.win.tue.nl/declare/.

Dijkman, R.M., Dumas, M., & Ouyang, C. (2008). Semantics and analysis of business process models in BPMN. *Information and Software Technology*, *50*(12), 1281–1294.

El Kharbili, M. (2012). Business process regulatory compliance management solution frameworks: a comparative evaluation. In *APCCM 2012, CRPIT* (Vol. 130, pp. 23–32).

Elgammal, A., Turetken, O., Heuvel, W.J., & Papazoglou, M. (2010). Root-cause analysis of design-time compliance violations on the basis of property patterns. In P. Maglio, M. Weske, J. Yang, & M. Fantinato (Eds.), *Service-oriented computing, lecture notes in computer science.* (Vol. 6470, pp. 17–31). Berlin Heidelberg: Springer. doi:10.1007/978-3-642-17358-5_2.

Elgammal, A., Turetken, O., van den Heuvel, W.J., & Papazoglou, M. (2011). On the formal specification of regulatory compliance: a comparative analysis. In *Proceedings of ICSOC'10* (pp. 27–38).

Elgammal, A., Oktay, T., & Heuvel, W.J. (2012). Using patterns for the analysis and resolution of compliance violations. *International Journal of Cooperative Information Systems*, *21*(31). doi:10.1142/S0218843012400023.

Elgammal, A., Turetken, O., van den Heuvel, W.J., & Papazoglou, M. (2014). Formalizing and applying compliance patterns for business process compliance. *Software & Systems Modeling*, 1–28. doi:10.1007/s10270-014-0395-3.

Fellmann, M., & Zasada, A. (2014). State-of-the-art of business process compliance approaches. In *22st European conference on information systems, ECIS 2014, Tel Aviv, Israel, June 9-11, 2014*. http://aisel.aisnet.org/ecis2014/proceedings/track06/8.

Gambini, M., Rosa, M., Migliorini, S., & Hofstede, A.H.M. (2011). Automated error correction of business process models. In S. Rinderle-Ma, F. Toumani, & K. Wolf (Eds.), *Business process management, LNCS* (Vol. 6896, pp. 148–165). Berlin Heidelberg: Springer.

Ghose, A., & Koliadis, G. (2007). Auditing business process compliance. In B. Krämer, K.J. Lin, & P. Narasimhan (Eds.), *Service-oriented computing (ICSOC 2007), LNCS* (Vol. 4749, pp. 169–180). New York: Springer. doi:10.1007/978-3-540-74974-5_14.

Goedertier, S., & Vanthienen, J. (2006). Designing compliant business processes with obligations and permissions. In J. Eder & S. Dustdar (Eds.), *Business process management workshops, lecture notes in computer science* (Vol. 4103, pp. 5–14). Berlin Heidelberg: Springer. doi:10.1007/11837862_2.

Gordon, T.F., Governatori, G., & Rotolo, A. (2009). Rules and norms: requirements for rule interchange languages in the legal domain. In *RuleML 2009, LNCS* (Vol. 5858, pp. 282–296). Springer.

Governatori, G. (2005). Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, *14*(2-3), 181–216. doi:10.1142/S0218843005001092.

Governatori, G. (2015). Thou Shalt is not you will. In *Proceedings of the 15th international conference on artificial intelligence and law (ICAIL 2015)*. ACM. doi:10.1145/2746090.2746105.

Governatori, G., & Rotolo, A. (2010a). A conceptually rich model of business process compliance. In *Proceedings of APCCM '10*. (Vol. 110, pp. 3–12).

Governatori, G., & Rotolo, A. (2010b). Norm compliance in business process modeling. In *RuleML 2010: 4th international web rule symposium* (pp. 194–209). Springer. doi:10.1007/978-3-642-16289-3_17.

Governatori, G., & Sadiq, S. (2009). The journey to business process compliance. In *Handbook of research on business process management, IGI Global* (pp. 426–454).

Governatori, G., Hoffmann, J., Sadiq, S.W., & Weber, I. (2008). Detecting regulatory compliance for business process models through semantic annotations. In *Business process management workshops'08* (pp. 5–17).

Hashmi, M., & Governatori, G. (2013). A methodological evaluation of business process compliance management frameworks. In M. Song, M. Wynn, & J. Liu (Eds.), *Asia pacific business process management, LNBIP* (Vol. 159, pp. 106–115). Switzerland: Springer.

Hashmi, M., Governatori, G., & Wynn, M.T. (2012). Business process data compliance. In *Rules on the web: research and applications - 6th international symposium, RuleML 2012, Montpellier, France, August 27-29, 2012.* Proceedings (pp. 32–46). doi:10.1007/978-3-642-32689-9_4.

Hashmi, M., Governatori, G., & Wynn, M.T. (2013). Normative requirements for business process compliance. In *Service research and innovation - third Australian Symposium, ASSRI 2013, Sydney, NSW, Australia, November 27-29, 2013, revised selected papers* (pp. 100–116). doi:10.1007/978-3-319-07950-9_8.

Hashmi, M., Governatori, G., & Wynn, M.T. (2014). Modeling obligations with event-calculus. In *Rules on the web. From theory to applications - 8th International Symposium, RuleML 2014, Prague, Czech Republic, August 18-20, 2014.* Proceedings (pp. 296–310). doi:10.1007/978-3-319-09870-8_22.

Hee, K., Hidders, J., Houben, G.J., Paredaens, J., & Thiran, P. (2010). On-the-fly auditing of business processes. In K. Jensen, S. Donatelli, & M. Koutny (Eds.), *Transactions on Petri nets and other models of concurrency IV, LNCS* (Vol. 6550, pp. 144–173). New York: Springer.

Herrestad, H. (1991). Norms and formalization. In: ICAIL'91, ACM, (pp. 175–184). doi:10.1145/112646.112667.

Hinge, K., Ghose, A., & Koliadis, G. (2009). Process SEER: A Tool for Semantic Effect Annotation of Business Process Models. In *EDOC '09. IEEE international* (pp. 54–63). doi:10.1109/EDOC.2009.24.

Hoffmann, J., Weber, I., & Governatori, G. (2012). On compliance checking for clausal constraints in annotated process models. *Information Systems Frontiers*, *14*(2), 155–177.

Jiang, J., Dignum, V., Aldewereld, H., Dignum, F., & Tan, Y.H. (2013). Norm compliance checking. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, international foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS '13* (pp. 1121–1122). http://dl.acm.org/citation.cfm?id=2484920.2485101.

Jiang, J., Aldewereld, H., Dignum, V., Wang, S., & Baida, Z. (2014). Regulatory Compliance Of Business Processes. AI & SOCIETY, (pp. 1–10). doi:10.1007/s00146-014-0536-9.

Kiepuszewski, B., Hofstede, A.H.Mt., & Bussler, C. (2000). On structured workflow modeling. In *Proceedings of the 12th international conference on advanced information systems engineering, CAiSE '00* (pp. 431–445). London: Springer.

Letia, I.A., & Groza, A. (2013). Compliance checking of integrated business processes. *Data & Knowledge Engineering*, *87*(0), 1–18. doi:10.1016/j.datak.2013.03.002.

Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Goeser, K., Reichert, M., & Dadam, P. (2010). SeaFlows toolset - compliance verification Made Easy. In *CAiSE'10 Demos*.

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

Ly, L.T., Maggi, F.M., Montali, M., Rinderle, S., & vanvan der Aalst, W. (2013). A framework for the systematic comparison and evaluation of compliance monitoring approaches. In *Proceeding of EDOC*.

Maggi, F., Montali, M., Westergaard, M., & van der Aalst, W. (2011). Monitoring business constraints with linear temporal logic: an approach based on coloured automata. In *BPM, LNCS 6896* (pp. 132–147). Springer.

Mulo, E., Zdun, U., & Dustdar, S. (2013). Domain-specific language for event-based compliance monitoring in process-driven soas. *Service Oriented Computing and Applications*, *7*(1), 59–73. doi:10.1007/s11761-012-0121-3.

Murata, T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.

Orriëns, B., Yang, J., & Papazoglou, M.P. (2003). A framework for business rule driven service composition. In B. Benatallah, & M.-C. Shan (Eds.), *Technologies for e-services*, *lecture notes in computer science* (Vol. 2819, pp. 14–27). Berlin Heidelberg: Springer. doi:10.1007/978-3-540-39406-8_2.

Ouyang, C., Dumas, M., Breutel, S., & ter Hofstede A.H.M. (2006). Translating Standard Process Models to BPEL. In *CAiSE* (pp. 417–432).

Ouyang, C., Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., & Mendling, J. (2009). From business process models to process-oriented software systems. *ACM Trans Softw Eng Methodol, 19*(1).

Pesic, M., & van der Aalst, W.M.P. (2006). A declarative approach for flexible business processes management. In J. Eder, & S. Dustdar (Eds.), *Business process management workshops, lecture notes in computer science* (Vol. 4103, pp. 169–180). Berlin Heidelberg: Springer. doi:10.1007/11837862_18.

Ramezani, E., Fahland, D., van der Werf, J., & Mattheis, P. (2012). Separating compliance management and business process management. In F. Daniel, K. Barkaoui, & S. Dustdar (Eds.), *Business process management workshops, lecture notes in business information processing* (Vol. 100, pp. 459–464). Berlin Heidelberg: Springer. doi:10.1007/978-3-642-28115-0_43.

Ramezani, E., Fahland, D., van Dongen, B.F., & van der Aalst, W.M.P. (2013). Diagnostic information for compliance checking of temporal compliance requirements. In *CAiSE* (pp. 304–320).

Rodrìguez, C., Schleicher, D., Daniel, F., Casati, F., Leymann, F., & Wagner, S. (2013). Soa-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes. *Service Oriented Computing and Applications*, *7*(4), 275–292. doi:10.1007/s11761-013-0129-3.

Sadiq, S., Governatori, G., & Namiri, K. (2007). Modeling control objectives for business process compliance. In *Proceedings of BPM'07* (pp. 149–164). Springer. http://portal.acm.org/citation.cfm?id=1793114.1793130.

Sartor, G. (2005). *Legal reasoning: a cognitive approach to the law.* Springer.

Turki, S., & Bjekovic-Obradovic, M. (2010). Compliance in e-government service engineering: state-of-the-art. In *Exploring services science, LNBIP* (pp. 270–275). Springer.

van der Aalst, W.M.P. (1998). The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, *8*(1), 21–66.

van der Aalst, W.M.P. (2000). Workflow verification: finding control-flow errors using petri-net-based techniques. In W.M.P. van der Aalst, J. Desel, & A. Oberweis (Eds.), *Business process management: models, techniques, and empirical studies*.

van der Aalst, W., Adriansyah, A., & van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis. *Wiley Int Rev Data Min and Knowl Disc*, *2*(2), 182–192.

Weigand, H., van den Heuvel, W.J., & Hiel, M. (2011). Business policy compliance in service-oriented systems. *Information Systems*, *36*(4), 791–807.

Wen, L., Wang, J., van der Aalst, W.M., Huang, B., & Sun, J. (2010). Mining process models with prime invisible tasks. *Data & Knowledge Engineering*, *69*(10), 999–1021.

Mustafa Hashmi is a graduate researcher with NICTA Queensland Research Laboratory, Australia. Mustafa received the Bachelor Degree (CS) in 2000 from Pakistan and Masters Degree (IT) in 2004 from Malaysia. Currently, he is pursuing his PhD at Queensland University of Technology (QUT), Australia. His research interests are in the area of automation and analysis of business processes, business process compliance management,normative systems, legal knowledge representation, defeasible Logic and defeasible reasoning, non-monotonic reasoning and their applications to solve complex problems in large scale enterprises. Since 2011, Mustafa is a member of the technical committee on OASIS Legal RuleML.

Prof. Guido Governatori received the PhD degree in legal informatics from the University of Bologna, Italy. He is a senior principal researcher in the Software Systems Research Group at NICTA, where he leads the research activities on business process compliance. He is also an adjoint professor in the BPM Group at Queensland University of Technology (QUT), Australia. His research interests include defeasible reasoning, modal deontic and non-classical logics and their applications to normative reasoning, agent systems, and business process modelling. Prof. Governatori is an editor of the deontic logic corner of the Journal of Logic and Computation and the agent and norms section of Artificial Intelligence and Law journal.

Dr. Moe Thandar Wynn is a senior lecturer within the Business Process Management Discipline at Queensland University of Technology (QUT), Brisbane, Australia. She holds a PhD in the area of workflow management from Queensland University of Technology (2007). Her research interests include: Business process modelling, process automation, business process improvement, and process mining. She has over 60 publications in journals and international conferences in the BPM area. Dr. Wynn has secured over 2.4 Million AUD dollars in competitive grant funding as a chief investigator for 6 grants in the last five years. Dr. Wynn is a member of IEEE taskforce on Process Mining and a working group member for IEEE standardization of XES log standard. Since 2011, she has been involved as a researcher within the NICTA Queensland Research Laboratory.