


Norms modeling constructs of business process compliance management frameworks: a conceptual evaluation

Mustafa Hashmi¹  · Guido Governatori¹ 

© Springer Science+Business Media B.V. 2017

Abstract The effectiveness of a compliance management framework (CMF) can be guaranteed only if the framework is based on sound conceptual and formal foundations. In particular, the formal language used in the CMF is able to expressively represent the specifications of normative requirements (hereafter, norms) that impose constraints on various activities of a business process. However, if the language used lacks expressiveness and the modelling constructs proposed in the CMF are not able to properly represent different types of norms, it can significantly impede the reliability of the compliance results produced by the CMF. This paper investigates whether existing CMFs are able to provide reasoning and modeling support for various types of normative requirements by evaluating the conceptual foundations of the modeling constructs that existing CMFs use to represent a specific type of norm. The evaluation results portray somewhat a *bleak picture of the state-of-the-affairs* when it comes to represent norms as none of the existing CMFs is able to provide a comprehensive reasoning and modeling support. Also, it points to the shortcomings of the CMFs and emphasises exigent need of new modeling languages with sound theoretical and formal foundations for representing legal norms.

Keywords Norms · Compliance · Business processes · Modelling constructs · Modelling languages · Compliance management frameworks

✉ Mustafa Hashmi
mustafa.hashmi@data61.csiro.au

Guido Governatori
guido.governatori@data61.csiro.au

¹ Data61, CSIRO, 41 Boggo Road Dutton Park, Brisbane, QLD 4102, Australia

1 Background

The fall of corporate giants such as Enron, WorldCom, American International Group (AIG) in the US and Parmalat in Europe caused a severe depression in the world's financial markets. The apparent deterioration in the reliability of information as the result of bad management, the declaration of insufficient and false reporting of an organisation's values, and uncontrolled transfer of money are some of the reasons given for this depression. Fongon and Grillo (2004) observed that bad governance, unreliable information and non-conformance to the regulatory laws resulted in a USD 5 trillion loss to organisations between March 2000 and September 2002 alone. The collapse and ultimate closures of large companies resulted in the urgent need to design and implement new regulatory laws to control that how businesses should conduct their operations in future. Thus, several laws such as Sarbanes–Oxley (SOX) Act (US-Government 2002); BASEL (series of) Acts (SCBS 2004; BCBS 2013); Health Insurance Portability and Accountability Act (HIPAA 1996), and Anti-Money Laundering regulations and monetary *de-facto* standards (such as the International Financial Reporting standard [IFRS] (IFRS 2014), Financial Action Task Force [FATF] recommendations (FATF 2017) etc.) have emerged. These had a direct impact on the operations of an organisation. Failure to comply with these regulatory laws can damage investors confidence, and result in financial penalties or (even) criminal prosecutions. Hence, adherence to the regulatory laws, internal controls, and other sources of compliance requirements has become a *must-to-do* activity for every organisation in the interests of transparency and more efficient operations of their daily business (Abdullah et al. 2010).

The term '*compliance*' is generally used in a very broad sense denoting the adherence to *source rules* (developed to apply to participants and on their behaviour in some local contexts) to the *target rules* (developed for a wider community of participants in a potentially global context). Essentially, target rules such as norms, laws, regulations, recommendations etc., prescribe some requirements or expectations—and if acted upon, generate some effects or value to their subjects. Adherence of source rules to the target rules guarantees that prescribed requirements or expectations can be met (Governatori and Sadiq 2009). From a business process compliance perspective, compliance aims to ensure that organisations' operations, business processes and practices are in alignment with a prescribed set of regulatory laws. McIntyre (2008) defines compliance as:

A desired outcome, with regard to law and regulations, internal policies and procedures, and commitments to stakeholders that can be consistently achieved through a managed investment of time and resources. The compliance management includes the legal and tactical activities in day-to-day business processes.

In contrast, Governatori (2005) views *compliance* as:

an act/process to ensure that business operations, processes, and practices are in accordance with prescriptive (often legal) documents.

From the above definitions, it is clear that the term *compliance* connects two distinct domains: the *business process domain* and the *legal domain*. Each of these domains has its own specificities, which are designed to achieve specific goals. The business process domain is more *descriptive* detailing *how* the business activities should be performed. A business process can be seen as a self-contained temporal, and logical order in which a set of activities (called *tasks*) are performed to achieve a specific business objective. The business process domain is a well-researched and established domain, where researchers predominantly focus on developing and improving business process modeling approaches and languages to achieve flexibility in the business process execution (Lu and Sadiq 2007; Johansson et al. 2012; Becker and Laue 2012; Mili et al. 2010). In contrast, the legal (that is, regulatory) domain is *prescriptive* in nature; it ascribes the legal boundaries by imposing conditions that detail which actions can be considered legal and which actions must be avoided during the execution of business process to stay compliant. Accordingly, compliance aims to gain more understanding *how* organisations *should* operate in a more sustainable way to continue providing their services while strictly observing all the applicable regulations that can significantly affect their business operations (Olivieri 2014).

In this paper we will provide an evaluation of compliance management frameworks designed to determine if an (existing) business processes of organisations complies with given regulations. In the rest of this section we will set out the parameters for the evaluation.

1.1 The problem

Enterprises model business processes to get an abstract view of their business operations, specifically of how they achieve their business goals and implement applicable regulatory laws their business is subject to. Hence, business processes are the natural place to verify the effectiveness of applicable regulations and internal control objectives (Karagiannis 2008). Currently, enterprises employ a number of business process compliance checking strategies: at *design-time* where analysts can check for compliant/non-compliant behaviour while processes are still being designed; at *runtime* processes are consistently monitored; and finally, log files are *audited* after the processes have been fully executed for any non-compliant issues (see Governatori and Sadiq (2009) for details on these strategies). To support organisations' compliance efforts, a plethora of compliance management frameworks (hereafter, CMFs) and compliance checking approaches offering various functional and operational capabilities exists (see Becker et al. (2012) for a list of existing approaches). Mostly, these CMFs support specific compliance requirements and domains and claim to provide full compliance management support. However, since the scope of norms is to identify whether a business process complies with the relevant regulations, the reasoning support all types of norms is imperative for automated compliance checking of business processes.

From a business process compliance perspective, the main problem is to ensure whether the activities performed during the execution of the process are in alignment with the specifications of norms controlling the process behaviour.

Formalising the normative specifications of compliance rules using some formalism enables an automated verification of normative specifications over business process specifications. Essentially, from a formal representational perspective, most of the formalisms are able to represent the specifications of norms and that of business processes. For example, temporal logic is able to formally represent the specifications of a business process; that is, the sequence of states corresponding to the tasks of a business process (Governatori 2015). The authors in Hashmi et al. (2013) presented a classification of norms built upon the generic temporal model related to the persistence effects of legal norms (Palmirani et al. 2011), and provided the semantics definitions of the classes of obligations based on temporal validity of the norms and their effects. This raises the question whether existing formalisms can faithfully represent different types of norms such as obligations, permissions etc—especially the classes of the classification model in an expressive and conceptually sound way.

As different norms classes have different meanings and different properties, they might introduce varying degree of complexities for a modeling language. Capturing the real meanings and nuances of a specific type of a norm is paramount for effective compliance checking. In particular the modelling language should specify what it means to comply with or violate a given norm, what are the consequences of such behaviours, and what are the relationships among the different notions or types of norms represented by the language (Hashmi et al. 2014). The failure to properly capture such nuances by a chosen formalism can significantly impede the ability of the proposed technique to validate the specifications of norms, thus can put a question-mark on the reliability of a CMF. In this paper we investigate whether existing CMFs are able to provide modeling support for all classes of the classification model—in particular, which modeling constructs they provide to model a specific class of the norms. We strongly believe that addressing this question is of utmost importance before adopting a formalism to propose a compliance checking technique in the framework for real-life cases of norms. Therefore, unless we are certain or have a positive answer to this question, the effectiveness of a CMF based on some formalism for representing and checking the compliance of different types of norms will be pointless. Notice that, a CMF might address one or more aspects such as modeling compliance requirements, compliance checking, verification, enforcement, monitoring, conflict resolution and violation handling, reporting, to name but a few. In this paper, along the legal component of the compliance problem, we only focus on the norms modeling aspect which has direct impact on the effectiveness of a CMF because translating the compliance requirements into a machine-processable format is the foremost task for automated verification of the compliance requirements.¹

¹ Similarly, a CMF can have several modules and features that might have impact on the effectiveness of the CMF. In this paper, we concentrate only on their ability to correctly represent the legal requirements. If a CMF is not able to properly represent some types of requirements, then human intervention is needed to rectify such situations. This means that, potentially, compliance professionals have to check every single assessment done by the CMF, and nullifying thus the gain in efficiency of using automated support tools for compliance checking.

Contribution This paper contributes a methodology and detailed evaluation of the selected CMFs using a conceptual model for representing the legal norms (Sect. 2) to examine their conceptual foundations. Mainly, the focus of this evaluation is on *one-to-one mapping* between the classes of the classification models and the modeling constructs existing CMFs use to represent different types of norms using a formal language for their compliance checking. In this paper, we do not investigate how the specifications of norms are modelled in a CMF instead we focus on the *conceptual foundations and the modeling constructs used in a CMFs to model a specific type of norm*.

1.2 Approach

The aim of this study is to investigate *what level of support* existing CMFs offer for the modeling of (legal) normative requirements. To this end, we employed a four-steps approach:

- The objectives for this study were defined in the first step,
- A set of evaluation criteria meeting evaluation objectives was then determined,
- A range of CMFs were selected for the evaluations, and
- One-to-one correspondence between norms modeling constructs of the selected CMFs and a specific class of norms was examined in the last step.

Figure 1 illustrates the overall evaluation approach.

Evaluation Objectives The main objective of the conceptual evaluation was to examine the conceptual foundations of existing CMFs. We specifically looked at the conceptual approach that a framework proposes for checking the compliance of business processes, and the reasoning support for the norms representing various types of obligations. More specifically, the goals of the conceptual evaluation were to determine:

1. *the compliance checking approach used in a framework;*
2. *what constructs are provided for modeling the norms, and the formal language for doing so;*
3. *how the norms are linked to business processes for compliance checking;*

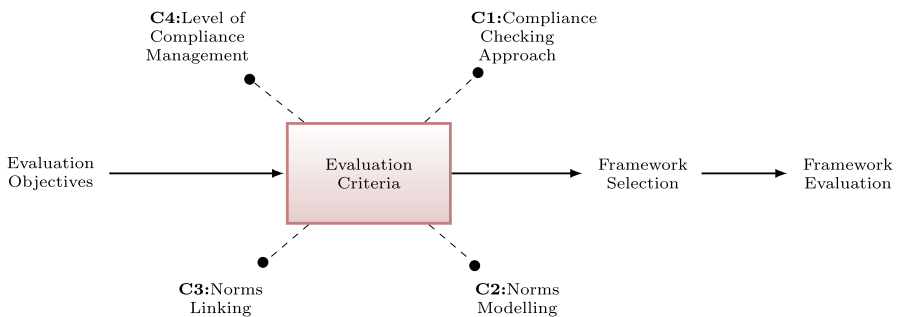


Fig. 1 Conceptual evaluation approach

4. *the level of compliance management support—that is, the framework contribution type, and whether all obligation modalities can be supported.*

Evaluation Criteria We defined a four-step selection criteria to identify representative frameworks namely: *compliance checking approach*; *norms modeling*; *norms linking*; and *level of compliance management*.

1. *The Compliance checking approach* This criterion is divided into two dimensions namely (a) *process lifecycle* aspects such as control-flow, resource, time etc., and (b) *orientation* of the compliance checking approach. The lifecycle aspect aims to examine whether the proposed approach in a CMF is design-time, run-time or post-execution time compliance checking. The orientation dimension, on the other hand, looks at whether the approach in a CMF is focused on the verification (or validation), or is purely business oriented. We use this criterion because of the focus of this paper is on design-time CMFs.
2. *Norms modeling* This criterion enabled us to examine *how CMFs model different types of norms (compliance requirements) and the formal logic they use to do so*. Specifically, we used this criterion to identify the modeling constructs for a specific obligation modality proposed in a CMF, and the ways in which it is modelled. For this purpose, we followed the approach used in van der Aalst et al. (2002) where the authors evaluated workflow patterns by describing the conditions that should hold for a pattern to be applicable under real-life situations. For our purpose, we also aim to examine whether the constructs proposed in a CMF can fully capture the compliance rules from the real-life regulations. Hence, our recourse is to check *one-to-one* mappings between the constructs provided in a CMF and the classes of norms to determine whether the construct can fully represent the intuition of the obligation modalities. Notice that the obligations modalities discussed in Sect. 2 describe temporal properties of obligations with respect to their validity and the effects of violations i.e., when an obligation enters into force, until what time it remains in force and the time of application. The proposed model is agnostic about the nature of the contents of the normative requirements (whether they express constraints about the data, resource, time or control-flow of the business process).
3. *Norms linking* Using this criterion, we identify *how a CMF links compliance rules with business process models*. The reason for using this criterion was to verify the compliance of normative requirements needed to provide both the formal model of norms and the formal model of a business process. Furthermore, if both models are represented in different formal languages, then a bridging mechanism serving as an interface between the formalisation of the business processes and the formalisation of norms is mandatory. For effective compliance checking of the compliance rules, it is imperative that they are properly modelled and linked to the business processes; if not, the results of the compliance checking approaches might not reliable. Hence, with this criterion, we aimed to look at the ways in which existing CMFs link the

compliance requirements to ensure that the compliance checking process remains transparent to the stakeholders.

4. *Level of compliance management* This criterion describes the level of support a CMF provides; that is, modeling, linking, compliance checking, and handling violations of the norms. Only CMFs that provide full compliance management support were selected. Those that merely provide a compliance checking algorithm or a modeling language were not considered.

Sample Framework Selection Although we reviewed and analysed a large number of CMFs (19 in total) (such as MASTER framework (MASTER 2008); REO-Toolkit (Arbab 2004); COMPAS (Elgammal et al. 2011); SeaFlows (Ly et al. 2010b, 2012); REALM (Giblin et al. 2005); NORMC (Kaźmierczak et al. 2012); and PSA@R (Rieke et al. 2014) to name but a few), and 45 compliance checking approaches (such as static compliance checking, logic-based, and pattern-based approaches), we refrained from undertaking a systematic literature survey, as done in Abdullah et al. (2010), Becker et al. (2012) and El Kharbili (2012). Since most of the existing CMFs are either prototypical implementations at very early stages of their development or they are mainly used for research purposes, no real data is available on how enterprises are using these CMFs in practice. Thus, we had to rely on the expert discussions that entail discussing various features that a CMF should have as pointed in El Kharbili (2012) within our research group. In addition, we supplemented the expert discussions by considering the CMFs those mostly cited in literature to make the evaluations more relevant. The term *most cited* frameworks reflects the traction that a CMF received at the time of the writing this paper.² This allowed us to start the evaluations with minimal information available on the CMFs. As a first step, we conducted an extensive literature search using the approach from Bandara et al. (2011), and along the regulatory compliance management frameworks dimensions listed in El Kharbili (2012). To search the resources, we used keywords-based search on these dimensions such as compliance rules modeling; verification; violation management; temporal rules; and also the keywords closely related to the evaluation criteria. This resulted in the extraction of more than 100 articles covering a range of compliance dimensions. Not all articles were relevant to the evaluation objectives, and unrelated articles were safely discarded. Only articles that contained key terms such as design-time, run-time, compliance framework, compliance checking approach were selected. Based on these, we selected *seven frameworks* meeting the evaluation criteria namely: PCL (Sadiq et al. 2007), BPMN-Q (Awad et al. 2008a), PENELOPE (Goedertier and Vanthienen 2006b), and COMPAS (Elgammal et al. 2011) as illustrated in Table 1. We believe that the selected CMFs give a fair representation of the most of compliance approaches, for example, design-time, run- and post-execution based approaches.

Norms Constructs Evaluation In the last step, we evaluated the norms modeling constructs of the selected CMFs for which we used the classes of classification

² Notice that, there might be several other works that have been published but do not received enough citations at the time of writing of this paper. We only consider CMFs based on Google Scholar citations between December 2013 and February 2017.

Table 1 Articles and # of citations of selected CMFs between Dec 2013–Feb 2017

Framework	References	2013	2017	Difference
PCL	Sadiq et al. (2007)	269	396	127
	Governatori et al. (2006)	209	270	61
	Lu et al. (2007)	74	115	41
	Governatori and Sadiq (2009)	62	107	45
	Governatori and Rotolo (2010)	29	54	35
	Sadiq and Governatori (2015)	28	71	43
BPMN-Q	Awad et al. (2008a)	159	246	87
	Awad (2007)	95	151	56
	Awad et al. (2008b)	48	80	32
	Awad et al. (2011)	38	78	40
SEAFLOWS	Ly et al. (2012)	64	117	53
	Ly et al. (2010a)	51	79	28
	Ly et al. (2010b)	44	69	25
	Ly et al. (2011)	43	84	41
Auditing	Ghose and Koliadis (2007)	166	231	65
Framework	Hinge et al. (2009)	25	53	28
DECLARE	Aalst et al. (2009)	214	392	178
	Montali et al. (2010)	98	173	75
	Maggi et al. (2011a)	66	133	67
	Maggi et al. (2011b)	15	46	31
	Ramezani et al. (2012a)	13	67	54
	Ramezani et al. (2013)	2	19	17
	Ramezani et al. (2012b)	17	31	14
COMPAS	Elgammal et al. (2010)	30	53	23
	Schumm et al. (2010)	22	43	21
	Türetken et al. (2011)	11	36	25
	Türetken et al. (2012)	8	33	25
	Elgammal et al. (2014)	9	27	18
PENELOPE	Goedertier and Vanthienen (2006b)	131	189	58
	Goedertier and Vanthienen (2006a)	32	43	11

model (discussed in Sect. 2) and provided semantic definitions of norms. The semantic definitions capture the meanings of norms (obligations) in terms of validity of a norm (i.e., the applicability conditions), what constitutes a violation and the effects of violations—and they are independent of any specific formalism (Hashmi et al. 2016). We use the formal definitions as benchmark for the evaluation where we examine *one-to-one* correspondence between the modeling constructs of a CMF and the semantic definition of a specific class of norms.

Outline The paper is structured as follows—in Sect. 2 a short discussion on obligation modalities with real-life examples of norms from regulatory documents is

given. Then conceptual evaluation of norms modeling constructs proposed in seven CMF is presented in Sect. 3 followed by a short discussion on the shortcomings and challenges in Sect. 4. Related works are discussed in Sect. 5 before the paper is concluded with some final remarks and pointers for future work in Sect. 6.

2 Obligation modalities

From a legal reasoning perspective, norms have different properties that can be used to analyse different aspects of norms. For example, rules validity, rules semantics, defeasibility, normative effects to name but a few. Hashmi et al. (2013) presented a classification of norms analysed from temporal validity of norms and the effects of violations on other obligations as the main criteria of classification and analysis. Since norms have their lifespan and produce effects when they are applied, from a business process compliance perspective, normative effects of interest are the deontic effects; that is, obligations, permissions and prohibitions. The classification is based on the temporal validity aspects of obligations—that is, the time (a) *when an obligation holds*, (b) *when an obligation is fulfilled* and (c) *time of application* (Hashmi et al. 2014; Palmirani et al. 2011). Essentially, obligations have lifespan and do not hold indefinitely, depending on the conditions—an obligation is removed from a set of active obligations once the contents of the obligation have been achieved or the obligation is violated.

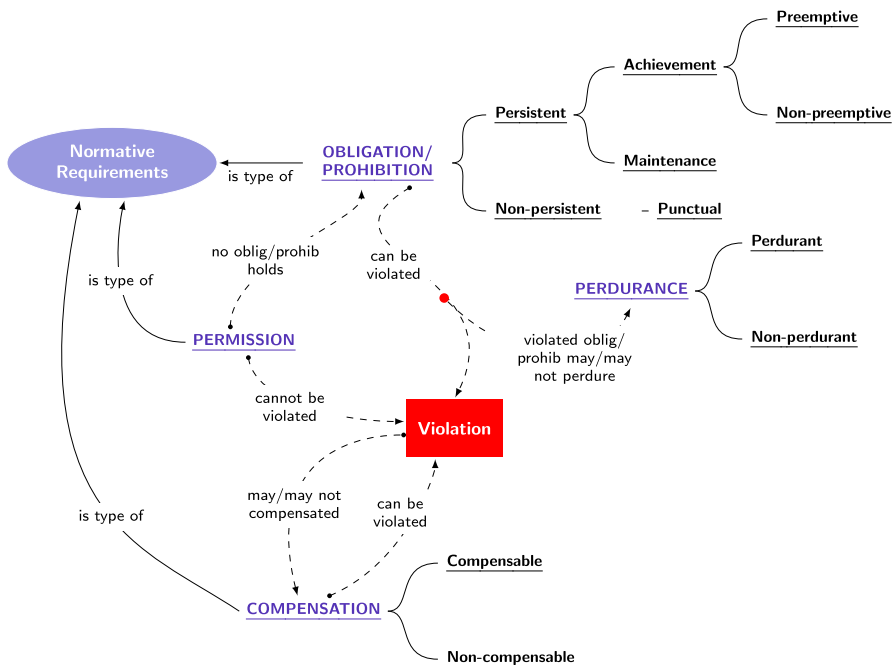


Fig. 2 Norms classes and possible relationship adopted from Hashmi et al. (2013)

Figure 2 depicts the basic deontic effects, and their relationship with the notion of violation and compensation. Besides the temporal aspects of the norms, the classification defines the effects of violations: namely, whether an obligation can be compensated upon a violation or the obligation still remains in force even after being violated. Essentially, *obligations* and *prohibitions* are constraints that restrict the behaviour of business processes. Obligations and prohibitions³ can be violated but permissions cannot. Thus, it seems that *permissions* do not play a direct role in compliance. This is not the case, since, they can be used as trigger to determine if other deontic effects are in force. Governatori (2015) gives an example of this phenomenon where not considering permissions leads to a wrong assessment whether a business process is compliant.

To determine the compliance of a process *whether it breaches a set of obligations*, the first step is to identify *whether* and *when* an obligation is in force. Since, obligations do not remain in force indefinitely, studying the lifespan of an obligation and its implications on the tasks of a process is important; hence, the question is *how long does an obligation hold for?* Generally, a norm can specify the applicability conditions for obligations, i.e., an obligation is in force for a particular time only—or indicates when an obligation enters into force. Typically, after an obligation enters in force it remains in force until it is terminated or removed, in this case we say we have a *persistent obligation*, otherwise if the obligation is in force only for a particular point in time, the obligation is said to be a *punctual obligation*.

For a punctual obligation the obligation must be fulfilled at the time the obligation is in force, this means that the content of the obligation must hold at that time, otherwise the obligation is violated. In contrast, persistent obligations are categories as *maintenance obligation* and *achievement obligation*. For a maintenance obligation, the conditions of the obligation must be fulfilled for all the instants in the interval in which the obligation holds. However, achieving the obligation contents for an achievement obligation *at least once* is sufficient. For an achievement obligation, if the obligation conditions are fulfilled even before obligation starts to hold, it is called a *preemptive obligation* otherwise, it is a *non-preemptive obligation*.

One property of the obligations is that they can be violated; however, the violation of an obligation does not necessarily mean that the process should be terminated. Depending on the violation conditions, a violated obligation can be compensated keeping the process compliant (Governatori and Sadiq 2009). For this purpose, legal contracts generally include compensatory provisions prescribing penalties and other sanctions that are enforced upon the breaches of obligation conditions. Essentially, not in all cases violations are compensable—hence, uncompensated violations result in the process being non-compliant or a penalty is imposed. Accordingly in certain cases, the effects of violation must be considered. If an obligation persists after violation, it is called a *perdurant obligation*—and otherwise it is a *non-perdurant obligation*.

³ Notice that, a prohibition is also called *negative obligation*, i.e., obligation not. Hence, whenever we consider obligation, we also include prohibitions (see Hashmi et al. 2014, for details).

In the coming sections, we use these classes (hereafter *deontic modalities*) as template to evaluate whether existing CMFs are able to represent the deontic modalities of the classification model, and which constructs they provide to represent a particular type of obligation. Notice that, the presented classification is complete in the sense that we consider temporal properties of a norm i.e., when an obligation enters into force, until what time it remains into force, and what effects it produces when violated.

Next, we describe in details each of these modalities, their semantics giving the conditions of applicability, and provide real-life examples from regulatory documents. In the rest of this section we are going to introduce the definitions of the normative concepts presented in Hashmi et al. (2016), and to discuss how they apply into the present setting. First, we formally define the meanings of *obligations*. To this end we introduce the *State* and *Force* functions required to provide the semantics definitions of the various deontic concepts. For this purpose, we need the notion of *timeline* i.e., a totally ordered discrete set of time points.⁴ In addition, we also assume there exists a logical language \mathcal{L} for writing the formulas to represent obligations and the environment.

Definition 1 (*State*) Given a timeline, we define a function $State : \mathbb{N} \rightarrow 2^{\mathcal{L}}$

The intuition behind the function *State* is to identify what formulas that are evaluated as true at the n -th time instant of a timeline.

Definition 2 (*Force*) Given a timeline, we define a function $Force : \mathbb{N} \rightarrow 2^{\mathcal{L}}$

The function *Force* is to identify the obligations in force at the n -th instant of time in a given timeline.

2.1 Persistent obligations

Generally, a *persistent* obligation needs to be obeyed for all time instances within the interval it is in force—such that an obligation remains into force until it is fulfilled or removed. The semantic definition of a persistent obligation as follows:

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

where o is an element of the language meant to represent obligation (or more precisely the contents of an obligation), and the function *Force* identifies the obligations in force at the n -th instant of time in a given timeline, and a *persistent obligation* is in force between n and m .

A persistent obligation can be classified into (a) *achievement* and (b) *maintenance* obligations.

Modality-1 (*Achievement*)

Description For an *achievement* obligation, achieving the contents of the obligation once is enough to fulfill it. Generally, an achievement obligation has a

⁴ A *timeline* can be infinite and isomorphic to the set of natural numbers. For our analysis; however, we can restrict to a set of natural numbers in case of a finite timeline (Hashmi et al. 2014).

deadline associated with it until which it has to be fulfilled in order to avoid violation.

Formal Definition o is an *achievement obligation* 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 is *fulfilled* if and only if $\exists k : n \leq k \leq m, o \in \text{State}(k)$.

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

- o is *preemptive* and $\forall k : k \leq m, o \notin \text{State}(k)$.
- o is *non-preemptive* and $\forall k : n \leq k \leq m, o \notin \text{State}(k)$.

where k is time instant in the given timeline at which achievement obligation enters into force.

Rule Example 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 clause 8.2.1.a.i.B is an *achievement obligation* since the clause gives a deadline to achieve it.

For an achievement obligation two further aspects need to be considered whether the obligation is *preemptive* or *non-preemptive*. This aspect gives the violation conditions for an achievement obligation, thus next we considered these sub-modalities.

Modality-2 (Preemptive Achievement)

Description A *preemptive achievement* obligation modality refers to an obligation that can be achieved even before the obligation actually enters into force.

Formal Definition o is *preemptive achievement* obligation 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 *preemptive achievement* obligation o is *fulfilled* if and only if $\exists k : k < n, o \in \text{State}(k)$.
- A *preemptive achievement* obligation is *violated* if and only if $\forall k \leq n : o \notin \text{State}(k)$.

Rule Example 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.

Clause (3) of the above rule example illustrates a preemptive obligation in the sense that it requires a copy of the contract document is given to debtor, but the obligation is not applicable if the creditor provided a copy of the contract document earlier under clause (2) of the section.

Modality-3 (*Non-preemptive Achievement*)

Description A *non-preemptive achievement* obligation modality refers to an obligation that can be discharged only after it enters into force.

Formal Definition o is a *non-preemptive achievement* obligation 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 *non-preemptive achievement* obligation o is *fulfilled* if and only if $\exists k : n \leq k \leq m, o \in \text{State}(k)$.

Contrary to a preemptive obligation, the obligation o must be in force at the time of fulfillment.

Rule Example The clause 8.2.1.a.i.B in the rule example for *Modality-1* giving the conditions for an achievement obligation is also a non-preemptive obligation. Because a complaint can be acknowledged only after it is received.

Modality-4 (*Maintenance*)

Description Contrary to an achievement, a *maintenance* obligation must be complied with for all the instants of the interval in which the obligation is in force. Furthermore, no deadline is required for the maintenance obligation to detect violation. The violation is automatically triggered if the obligation contents is not achieved at any time instant in the time interval, or simply after the obligation is in force in case there is no deadline. In addition, contrary to an achievement obligation, it is more meaningful for a maintenance obligation to remain in force indefinitely after it starts to hold.

Formal Definition o is a *maintenance obligation* 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 is *fulfilled* if and only if $\forall k : n \leq k \leq m, o \in \text{State}(k)$.
- a maintenance obligation o is *violated* if and only if $\exists k : n \leq k \leq m, o \notin \text{State}(k)$.

For the case of a *preemptive maintenance* obligation, o is *fulfilled* if and only if:

- $\forall k : n \leq k \leq m, o \in \text{State}(k)$.

Rule Example 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.

2.2 Non-persistent obligations

A *non-persistent* modality indicates that an obligation is in force at a particular time point only, or more, often, when an obligation enters into force. In case of a non-persistent, the contents of the obligation must be obeyed for the instance it is in force and classified as *punctual* obligations.

Modality-5 (*Punctual*)

Description By definition, for a *punctual* obligation the contents of the obligation must be immediately achieved otherwise a violation is triggered.

Formal Definition An obligation o is a punctual obligation if and only if:
 $\exists n \in \mathbb{N} : o \notin \text{Force}(n-1), o \notin \text{Force}(n+1), o \in \text{Force}(n)$.

- A punctual obligation o is *fulfilled* if and only if $o \in \text{State}(n)$.
- A punctual obligation o is *violated* if and only if $o \notin \text{State}(n)$.

Rule Example 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;

Clause 8.2.1.a.i. A stipulates a punctual obligation to acknowledge the complaint made either in person or by phone. Since, the obligation has to be achieved *immediately* upon reception of it—thus, it can be one of the activities done in the task ‘receive complaint’.

2.3 Modalities capturing effects of violations on obligations

One of the properties of obligations is that they can be violated. However, the violation of an obligation does not necessarily mean the imposition of a penalty (Hashmi et al. 2013). However, in legal reasoning norms may be associated with the conditions that take effects after the primary conditions of an obligation is violated.

Next, we discuss obligations modalities that capture various effects of the violation of an obligation. First we consider the *perdurance* effects of violations.

Modality-6 (*Perdurant*)

Description An obligation that *persists* even after being violated is considered as a perdurant obligation.⁵

Formal Definition o is a perdurant obligation 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 a deadline d in force between n and m is violated if and only if $\forall j, j \leq d, o \notin \text{State}(j)$.

Rule Example 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;

Essentially, Clauses 8.2.1.a.i.A and 8.2.1.a.i.B give the deadlines when to acknowledge a complaint respectively—however, Clause 8.2.1.a.i states that the complaint has to be acknowledged. No matter either clause A or B is violated (i.e., the complaint is not acknowledged within the deadline); the supplier still need to acknowledge the complaint. Hence, the obligation in clause 8.2.1.a.i is a perdurant obligation.

An obligation is considered non-perdurant, if it does not persist after the violation.

⁵ Notice that, all types of obligations have their own deadline (which is the end point of the interval in which an obligation is in force). The deadline can be used to indicate whether the obligation has been violated or not. Perdurant is a special case of obligation, which should have an explicit deadline which does not coincide with the end of the period when the obligation is in force.

Finally, we consider the modalities that capture the notion of *compensation* of a violation of an obligation. Such modalities are concerned with the contrary-to-duty obligations and fulfilling them make amend of the violation.

Definition 3 (*Compensation*) A compensation is a function $Comp : \mathcal{Q} \mapsto 2^{\mathcal{Q}}$

The idea of function *comp* is to associate to each formula a set of formulas, where a formula represents an obligation, and if the obligation is violated, then the obligation is compensated by the other formulas associated to the obligation. The formal definition in modality 7 captures the notion of compensation of the violation of an obligation.

Modality-7 (*Compensable*)

Description Depending on the violation conditions of an obligation, an obligation is *compensable*, if another obligation—or a set of obligations, enters into force to amend the violation of the primary obligation.⁶

Formal Definition o is compensable if and only if $Comp(o) \neq \emptyset$ and $\forall o' \in Comp(o)$,

$\exists n \in \mathbb{N} : o' \in Force(n)$.

o is compensated if and only if it is violated and $\forall o' \in Comp(o)$ either:

- o' is not violated;
- o' is compensated.

Rule Example-1 TCPC 2012. Article 8.1.1.

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

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

Rule Example-2 YAWL Deed of Assignment (Warranties & Indemnity), Clause 5.2⁷

⁶ For an obligation to be compensable a number of requirements must be followed, see formal definitions of *Compensable* and *compensation* for details on the $Comp(o)$ function (Hashmi et al. 2016).

⁷ <http://www.yawlfoundation.org/files/YAWLDeedOfAssignmentTemplate.pdf>, retrieved on March 28, 2013.

(5.1) Each Contributor warrants that:

5.1.1 the Intellectual Property assigned to the Foundation by the Contributor under *clause 2* comprises original works only, which have not been, and will not be, copied wholly or substantially from any other works,

...

5.1.3 it has the right to assign and grant the rights under *clause 2*.

(5.2) Each Contributor indemnifies and will defend the Foundation against any claim, liability, loss, damages, cost and expense 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 1 prescribes a compensation that the complaint must be escalated (or managed to supplier's internal escalation process), if it does not satisfy the conditions of clause (A), permitting the customer to ask for escalating the complaint to a higher level for resolution thus compensates the violation of clause (A). Essentially in this situation, the compensation obligation captures a sub-ideal but still acceptable behaviour. Contrarily, the obligation in clause 5.2 of rule example 2 obligates the contributor to compensate the YAWL foundation for any (negative) consequence sustained by the foundation if the contributor is in breach of clause 5.1. In this case the conditions of clause 5.2 impose a sanction (or penalty).

2.4 Additional deontic assignments

Modality-8 (*Permissions*)

Description *permissions* are special deontic effects allowing the bearer to perform some actions when there is no obligation or prohibition holds. Generally, permissions are not directly used for compliance instead their function is to determine that there are no obligations or prohibitions to the contrary. In addition, they can be used to derive other deontic effects. In legal theory, permission is considered as the absence of obligation to the contrary.

In deontic logic, the deontic operator for permission is **P** is assumed to be dual of the operator **O** for obligation (Makinson and van der Torre 2003). Thus, **P** and **O** satisfy the following equivalence relations:

$$O\phi \stackrel{def}{=} \neg P\neg\phi \quad \neg O\neg\phi \stackrel{def}{=} P\phi$$

The operator also follows the relationship $O\phi \longrightarrow P\phi$ meaning that if ϕ is obligatory, then ϕ is permitted (Sartor 2005). In this paper, we subscribe to this position for describing permissions.

Formal Definition **P** is permitted at time t if and only if $\neg P \notin Force(t)$.

Rule Example TCPC 2012. Article 6.2

Suppliers must undertake a Credit Assessment before providing a Post-Paid Service to a Consumer, and explain the financial implications of the provision of that Post-Paid Service to a Customer or their Guarantor.

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

- (a) Assess credit risk: undertake a Credit Assessment;
[...]
- (e) *Cooling off period for Guarantor* provide a Guarantor with a minimum 10 Working Days cooling off period in which the Guarantor may terminate the agreement for which the Guarantee is provided.

Modality-9 (Prohibitions)

Description A *prohibition* prescribes the conditions which a bearer should refrain from performing, a violation is triggered otherwise. Similar to permissions, legal theory assumes a strong relationship between obligation and prohibition—a prohibition is a ‘*negative obligation*’. Thus, something is *forbidden* if and only if we have obligation that *something* is not the case.

The equivalence relation between F and O forbidden and obligation operators as follows:

$$O\neg\phi \stackrel{def}{=} F\phi \qquad \neg P\phi \stackrel{def}{=} F\phi$$

meaning that if it is obligatory to do $\neg\phi$, then it is prohibited to do ϕ . Similarly, if ϕ is not permitted i.e., $\neg\phi$, then it is prohibited to do ϕ .

Formal Definition F is a prohibition in force if and only if $\neg F$ is a maintenance obligation in force. More specifically, prohibition corresponds to ‘*negative-maintenance*’ obligation.

Rule Example TCPC 2012. Article 7.12

- 7.12.0 If a Supplier proposes to move their Customers to an alternate wholesale network provider, the Supplier must notify all Customers being moved in the manner in which the Supplier normally communicates with them prior to that change being initiated. ...
- 7.12.3 Suppliers must not take any action that affects Telecommunications Services for which they are not the Supplier of that Telecommunications Service.

Generally, a prohibition implements a maintenance obligation, as shown in the above rule example. Essentially, it prescribes the conditions to refrain to start an activity until either a specific event occurs or a particular state is reached. According to Clause 7.12.3, Telco operators are not allowed to take any action that affects

Telecommunications Services for which they are not the Supplier of that Telecommunications Service.

3 Conceptual evaluation of norms modelling constructs

As was earlier discussed that various types of norms produce different effects when applied; thus, they may introduce different complexities for their compliance checking. Hence, a ‘one-size-fits-all’ modeling approach is far from being satisfactory from a conceptual point of view, and a CMF that does not represent the various nuances of obligations is not expressive enough. Hence, a conceptually weak CMF might not be suitable to provide any certification of compliance acceptable to accredited certifying organisations. In this section, we use the deontic modalities of the classification model discussed in the previous section as a template to examine the conceptual foundations of the selected frameworks especially by looking at *what constructs they provide to represent the classificatory classes of our classification model*. The first CMF evaluated is PENELOPE.

3.1 PENELOPE

PENELOPE (Goedertier and Vanthienen 2006b) is a formal language that declaratively captures legal constraints imposed on the enterprise’s business processes. The language verifies the compliant behaviour of business processes at design-time by using a proprietary algorithm that progressively enumerates all active obligations and permissions to generate the state-space and control flow. In a PENELOPE generated process, state-space is a set of all active obligations and permissions and conditional commitments that hold at a particular state. The flow of business interaction in PENELOPE processes is acyclic and moves from one state to the other. To compute the state space, the algorithm computes difference combinations of permissible actions that can be carried out until the deadline. Once all the states are enumerated, a BPMN model is drawn for all the parties involved in the interaction using IF-THEN rule. A task in the PENELOPE process represents a set of all obligations fulfilled by a partner in the interaction.

Whilst PENELOPE allows for representing the interaction between the parties, the violations in the generated process are represented using BPMN external events and exception constructs. The violation of an obligation or permission by a third party in a state, which cannot be repaired, is represented as an error end event. In contrast, the violation of a deadline of an obligation is modelled as intermediate time-out events. With the PENELOPE generated complaint processes different anomalies such as deontic and temporal conflicts and exceptions can be detected. The deontic assignments in PENELOPE are represented using well known event-based formalism Event-Calculus (EC) for modeling and reasoning with the legal norms. Table 2 shows the deontic properties proposed in PENELOPE modelled with EC. Next, we examine in details each of the deontic properties, and check whether these properties have some correspondence to the earlier discussed obligation modalities. Since the first three properties are related to the structure of the process

such as Choice (OR and XOR-Split), and Parallel (AND) gateways—we discard them from our analysis and directly evaluate the deontic properties.

Deontic Property-1 (*Obligation*)

Syntax $Oblig(\pi, \alpha, \delta)$

Description The term *Oblig* is used to capture the notion of obligations, where the syntax π refers to the agent (for example, an event or human resource) that triggers the obligation, α refers to the conditions (or actual contents) of the obligation, and δ is the deadline until which the obligation must be fulfilled.

Evaluation In combination with the *Initiates* predicate and *Terminates* predicate, the term can be used to model achievement obligation. The predicates *Initiates* and *Terminates* capture the effects when an obligation enters into force; and when it is terminated, respectively, depending on whether the obligation is fulfilled or removed. The term $Oblig(\pi, \alpha, \delta)$ defines the parameters of the agent who has to obey the contents of the obligation and fulfill it by a deadline.

Correspondence The term has one-to-one correspondence with the semantics of achievement obligations.

Deontic Property-2 (*Permission*)

Syntax $Perm(\pi, \alpha, \delta)$

Description This deontic property is used to capture the notion of permission, where in the syntax π refers to the agent the permission puts the constraint onto, α refers to the conditions (or actual contents) of obligation, and δ is the deadline until which the obligation must be fulfilled.

Evaluation Similar to property-1, the term *Perm* can be used to represent *permissions* in combination with the *Initiates* and *Terminates* predicates.

Table 2 Deontic properties of PENELOPE (Goedertier and Vanthienen 2006b)

Term	Meanings
$Oblig(\pi, \alpha, \delta)$	agent π must do the activity α by due date δ
$Perm(\pi, \alpha, \delta)$	agent π can do the activity α prior to due date δ
$CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2)$	agent π must do activity α_2 by due date δ_2 after activity α_1 is performed prior to the due date δ_1
(A) $Terminates(\alpha, Oblig(\pi, \alpha, \delta), \tau) \leftarrow \tau \leq \delta$	
(B) $Terminates(\alpha, Perm(\pi, \alpha, \delta), \tau) \leftarrow \tau \leq \delta$	
(C) $Happens(violation(Oblig(\pi, \alpha, \delta)), \delta) \leftarrow$ $ HoldsAt(Oblig(\pi, \alpha, \delta)) \wedge \sim Happens(\alpha, \delta)$	
(D) $Initiates(\alpha_1, Oblig(\pi, \alpha_2, \delta_2), \tau) \leftarrow$ $ \tau \leq \delta_1 \wedge HoldsAt(CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2)), \tau)$	

Correspondence One-to-one correspondence with *Permission*.

Deontic Property-3 (*Terminates Obligation/Permission*)

Syntax The syntax of the *Terminates* property for obligation and permission is as follows:

1. *terminates-obligation* $\text{Terminates}(\alpha, \text{Oblig}(\pi, \alpha, \delta), \tau) \leftarrow \tau \leq \delta$
2. *terminates-permission* $\text{Terminates}(\alpha, \text{Perm}(\pi, \alpha, \delta), \tau) \leftarrow \tau \leq \delta$

Description The meanings of the *Terminates* predicate is that the event α terminates the obligation fluent α at time τ .

Evaluation PENELOPE uses EC's *Terminates* predicate to terminate the effects of the normative notions of obligations and permissions. As mentioned earlier, obligations remain in force for a certain period of time and then they are removed—either upon fulfillment of violation. The *Terminates* predicate indicates when an obligation or permission ceases to hold.

Correspondence Though the predicate has no one-to-one mapping with the classes of obligations. However, in conjunction with *Initiates*, it can be used to capture the cessation of the effects of all obligations.

Deontic Property-4 (*Conditional Commitment*)

Syntax $CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2)$

Description PENELOPE uses the term *CC* to model conditional commitments (mostly in the sense of contractual interactions). The key idea behind the conditional commitments, as in most business scenarios and concrete applications, is that an agent commits himself to another agent to produce some effects when the antecedent of the conditional commitment holds. The meanings of the syntax is that an agent π must perform a certain activity (α_2) by the due date δ_2 , only after the agent has performed activity α_1 before the due date α_1 .

Evaluation Since conditional commitments are not considered in the classification model, this property is not evaluated. However, this deontic property can be useful from a structural compliance of a business process perspective, because a conditional commitment might represent the absence of the occurrence of an activity, for example, α_2 until another activity α_1 does not occur in the interaction.

Correspondence – NA –

Deontic Property-5 (*Initiates Conditional Commitment*)

Syntax The syntax for *Initiates* predicate for the conditional commitments is as follows:

$$\text{Initiates}(\alpha_1, \text{Oblig}(\pi, \alpha_2, \delta_2), \tau) \leftarrow \text{HoldsAt}(CC(\pi, \alpha_1, \delta_1, \alpha_2, \delta_2), \tau) \wedge \tau \leq \delta_1$$

Description The meanings of the syntax is that the occurrence of event α_1 *Initiates* the obligation for the agent π to do α_2 by the deadline δ_2 .

Evaluation There is only difference between this property and deontic property-4; that is, it gives the initiation conditions when a conditional commitment that an agent has to fulfill enters into force.

Correspondence – NA –

Handling the violations of obligations is one of the major requirements for a compliance framework as argued in (Awad 2010). Timely reporting of the violations not only allows the analysts to respond immediately but also saves much time and efforts. PENELOPE uses EC's *Happens* predicate to represent violations for which the framework provides the following semantic properties.

Deontic Property-6 *Happens* (Violation Handling)

Syntax The syntax for *Happens* predicate for the violations handling is as follows:

$$Happens(violation(Oblig(\pi, \alpha, \delta)), \tau) \leftarrow HoldsAt(Oblig(\pi, \alpha, \delta)) \wedge \sim Happens(\alpha, \delta)$$

Description The meaning of the syntax is that the obligation to do α by the deadline δ is violated at time τ when the obligation fluent α holds and the event fulfilling the obligation does not happen by the deadline.

Evaluation Violations in PENELOPE can only occur in the form of deadlock situations and temporal conflicts, we understand that the use of *Happens* predicate is useful from the capturing of the occurrence of events perspective. This is because EC *Happens* predicate can effectively reason about the events and the changes over time resulting from the occurrences of events over time. However, the violation semantics can only indicate when an obligation is violated. Hence, the semantics property can be used to indicate the violation of obligations but it cannot be used to reason about the effects the violation of an obligation might produce; for example, perdurance of the violated obligation or triggering of compensatory actions. Note that, since *preemptive achievement* obligations are fulfilled even before the obligation enters into force, they cannot be violated. Hence, no reasoning support is required for such obligation types.

Correspondence *Happens* predicate has correspondence with deontic pattern to determine the violation of an achievement obligation.

The deontic properties discussed above are used to model obligations, permissions and conditions commitments. However, other obligation modalities cannot be represented with these properties. This is because EC is not suitable for reasoning all types of obligations.⁸ Essentially, representing deontic properties for achievement, permission, and conditional commitments is only possible as deontic

⁸ A detailed evaluation of semantic properties, and the reasons why EC is not capable of providing a full reasoning and modeling support for all types of obligation, are discussed in (Hashmi et al. 2014).

property-4 and deontic property-5 can be used to determine when the obligations corresponding to the commitments enter into force. Also, these properties allow the explicit definition of deadlines as precedence rules. This is because the PENELOPE generates compliant processes from the rule set of obligations and permissions. The problem with PENELOPE framework is that it does not consider prohibitions under close world assumption in order to prevent inconsistencies that may arise due to incomplete information about all partners in the business interaction. While PENELOPE is able to model the deadlocks and temporal conflicts, it cannot provide the reasoning support for deontic conflicts. This is because prohibitions or waived obligations are not considered in PENELOPE. Table 9 illustrates the types of normative requirements that are supported by PENELOPE.

On the same note, it is not possible to handle the effects of the violations because the violation semantics discussed above can only indicate violation of the obligation. Because violation properties in PENELOPE are based on the notion of violations without reparation (VWR), a penalty is imposed whenever there is a violation of an obligation. No deontic properties are provided for capturing the effects of violations—that is, perdurance and compensatory actions; this is left to the analysts.

3.2 COMPAS

COMPAS⁹ (Elgammal et al. 2011) is a compliance management framework for SOA-based systems. Aiming to provide all-round compliance governance, the CMF is able to verify the compliance of business processes either at design-time, run-time or at evaluation time. For this purpose, it uses a model-driven approach for modeling compliant processes, which provides the ways to flexibly define the specifications of compliance concerns over business processes. The process models are designed using a view-based modeling framework, whereas domain specific language is used to model the compliance concern (Daniel et al. 2009).

At design-time, processes are annotated with compliance requirements that the process has to comply using a domain-specific language. The annotated processes, called fragments, are then stored into a process model repository. These re-usable process fragments are then fed into a code generator, which produce executable process models. These process models are then verified for any non-compliant issues at run-time. The compliance results produced by the dedicated enterprise service bus (ESB) are then saved in an event log. For auditing time evaluations, a special protocol component evaluates the stored event logs to verify whether the process models comply with all the constraints linked to the stored process fragment. In case the monitoring protocol identifies any non-compliance issues, a violation report is published as a violation event on a monitoring dashboard together with root-causes of the violation for remedial actions.

For compliance checking, business processes are annotated with compliance constraints in the form of (re-usable) process fragments. These fragments underline

⁹ COMPAS is an EU projects for Compliance-driven Models, Languages and Architectures for Services: <http://www.compas-cit.eu>.

the required behaviour of the control-flow of a process model, and they are formalised using linear temporal logic (LTL). Then the annotated process fragments are then assessed to validate the compliant behaviour of 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 behaviour 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.

As far as the modeling of compliance requirements is concerned, COMPAS uses a compliance request language (CRL) (Elgammal 2012; Elgammal et al. 2014) for modeling normative requirements. The core of CRL is its LTL-based graphical compliance patterns, which are high-level compliance templates to model the compliance constraints—predominantly, compliance requirements from the control flow (structural) perspective of business processes. In addition, most of these patterns are used by other frameworks, and more, recently additional patterns representing features specific to normative reasoning, such as exceptions to rules and compensation of violations have been included (Elgammal et al. 2014). The CRL graphical patterns representing different types of compliance rules are categorised into three distinct categories of patterns namely: *atomic patterns*, *resources patterns*, and *timed patterns*. The atomic patterns aim to describe the requirements involving the ordering of occurrence of the process elements, as depicted in Table 3. Some of the atomic patterns are based on Dwyer's property specification patterns (Dwyer et al. 1999), and categorised into *occurrence* and *ordering* patterns. Accordingly, *timed patterns* are concerned with the constraints that include temporal requirements, and *resources patterns* are related to constraints that are used to describe the recurring requirements pertaining to the the resources such as authorisation or task assignment constraints. For our purpose, we only consider patterns that are relevant to describe compliance rules based on the obligation modalities and discard those that are used to describe compliance constraints concerning the resources aspect of the business processes.

In addition to the atomic patterns, CRL defines *composite patterns*, as illustrated in Table 3. Composite patterns aim to describe more complex compliance rules, and are built conjunctively by combining multiple atomic patterns using Boolean logical operators such as NOT, AND, OR, XOR, Implies and Iff operators. These patterns are mapped into LTL formulas (see the description above) enabling the translation of CRL expressions into a set of LTL formulas. Essentially, these patterns are used to represent different types of compliance requirements.

Table 3 Atomic patterns and CRL expression (Elgammal et al. 2014)

Atomic pattern	Pattern-expressions	Description
isAbsent	ϕ isAbsent	ϕ should not exist throughout the process model
Exists	ϕ Exists	ϕ should occur at least once within the process model
BoundedExists	ϕ BoundedExists $\leq 2^*$	Shows that ϕ must occur at most 2 times within the process
	ϕ BoundedExists $\geq 2^*$	Shows that ϕ must occur at least 2 times within the process
isUniversal	ϕ isUniversal	P should always be true throughout the process
Precedes	ϕ Precedes ψ	ψ is always preceded by ϕ
Chain – Precedes	ϕ Precedes (σ, τ)	Meaning that a sequence of σ, τ must be preceded by ϕ
	(σ, τ) Precedes ϕ	ϕ must be preceded by a sequence of σ, τ
LeadsTo	ϕ LeadsTo ψ	ϕ must always be followed by ψ
Chain – LeadsTo	ϕ LeadsTo (σ, τ)	Shows that ϕ must be followed by a sequence of σ, τ
	(σ, τ) LeadsTo ϕ	Shows that a sequence of σ, τ must be followed by ϕ
Exists – Often	ϕ Exists – Often	ϕ must occur frequently within the process model
DirectlyFollowedBy	ϕ DirectlyFollowedBy ψ	Shows that required ϕ to be followed by ψ
Composite patterns		
ϕ CoExists ψ	$((\phi) \text{ Exists} \rightarrow (\psi) \text{ Exists})$	The presence of ϕ mandates that ψ is also present
ϕ CoAbsent ψ	$((\phi) \text{ isAbsent} \rightarrow (\psi) \text{ isAbsent})$	The presence of ϕ mandates that ψ is also absent
ϕ Exclusive ψ	$((\phi) \text{ Exists} \rightarrow (\psi \text{ isAbsent})) \wedge ((\psi) \text{ Exists} \rightarrow (\phi \text{ isAbsent}))$	The presence of ϕ mandates the absence of ψ , and presence of ψ mandates the absence of ϕ
ϕ Substitute ψ	$(\psi \text{ isAbsent} \rightarrow (\phi) \text{ Exists})$	ψ substitutes the absence of ϕ
ϕ Corequisite ψ	$(\phi) \text{ Exists} \text{ iff } (\psi) \text{ Exists} = ((\phi) \text{ Exists} \rightarrow (\psi) \text{ Exists}) \wedge ((\psi) \text{ Exists} \rightarrow (\phi) \text{ Exists})$	ϕ and ψ should either exist together or be absent together
ϕ MutexChoice ψ	$(\phi) \text{ Exists} \text{ xor } (\psi) \text{ Exists} = ((\phi) \text{ Exists} \wedge (\phi \text{ isAbsent})) \vee ((\psi) \text{ Exists} \wedge (\psi \text{ isAbsent}))$	Either ϕ or ψ exists but not any of them or both of them

We now discuss these patterns in detail, and examine whether they have correspondence with the obligation modalities.

Pattern-1 (*isAbsent*)

Description ϕ should not exist throughout the process model.

LTL Formula $G(\neg\phi)$

Evaluation As the name implies, **isAbsent** is used to indicate that some activity ϕ must not hold in the whole trace or throughout the business process model. This is because *prohibitions* specify the conditions that the bearer must avoid during the course of interaction. In contrast to achievement, prohibitions do not prescribe deadlines; that is, if a constraint prescribes the prohibition of some action, this must be maintained throughout the execution of the process. Hence, the pattern **isAbsent** can be used to express compliance constraints prohibiting some actions. The **isAbsent** pattern has a duality relation with **Existence** or **Eventually** patterns.

Correspondence The pattern has correspondence with *prohibition* modality.

Pattern-2 (*Exists*)

Description ϕ should occur at least once within the process model.

LTL Formula $F(\phi)$

Evaluation The **Exists** (or eventually) pattern is used to specify that the execution of the process model contains the instance of some proposition ϕ . The existence pattern has a strong relationship with the **Absence** pattern because of the duality relation; it can be used to specify the negation and explicit queries for the existence of some propositions. As for the definition of persistent obligations, an achievement obligation can specify the conditions that might eventually hold in future; thus, the pattern can be used to represent achievement obligation. Cases where the obligation specifies some additional conditions—that is, the occurrence of the proposition ϕ with some finite number of times—can be handled with the **Bounded Existence** patterns. COMPAS offers two **Bounded Existence** patterns for this purpose.

- The **BoundedExists** ($\phi\text{BoundedExists} \leq 2*$) shows that ϕ must occur at most 2 times within the process.

LTL Formula $\neg\phi W(\phi W(\neg\phi W(\phi W\neg F(\phi))))$

- The **BoundedExists** ($\phi\text{BoundedExists} \geq 2*$) specifies that the proposition ϕ must occurs at least 2 times within the process.

LTL Formula $\neg\phi W(\phi W(\neg\phi W(\phi)))$

¹⁰ The **BoundedExists** pattern can be useful for achievement, only if such conditions are prescribed by the norm. However, in every instance, an obligation can have independent conditions associated with it, with different objectives.

Correspondence The pattern can be used to represent *achievement* obligations.¹⁰

Pattern-3 (*isUniversal*)

Description ϕ should always be true throughout the process.

LTl Formula $G(\phi)$

Evaluation The pattern *isUniversal* (also called ‘Always’ or ‘Global’) aims to define the part of process execution that includes states in which the presence of proposition ϕ is always desired. As for the maintenance obligations, a special case of persistent obligations, the obligation conditions must hold for all instances of the interval in which the obligation is in force. The *isUniversal* pattern can be used to handle maintenance obligations. Accordingly, the pattern has a close relationship with the *Absence* and *Existence*, pattern and can be equally applied in situations where the *isAbsent* pattern can be applied. This is because the universal presence of some proposition ϕ can be seen as the absence of its negation; that is, $G(\neg\phi)$. Hence, it can also be suitable for representing cases of *prohibitions*.

Correspondence The pattern can be used to represent *maintenance* obligations and *prohibitions*.

Pattern-4 (*Precedes*)

Description indicates that activity ψ is always preceded by another activity ϕ .

LTl Formula $\neg\psi W\phi$

Evaluation The *Precedes* pattern (also called *precedence*) describes the relationship between two propositions; for example, ϕ and ψ where the execution of ϕ is mandatory for the execution of ψ . In other words, the execution of the first proposition enables the execution of the second proposition. A common norm example for precedence is: a payment can be made only after an invoice is issued; that is, the issuance of the invoice is a pre-condition for make payment event. The *Chain Precedes* is the variant of the *Precedes* pattern COMPAS uses to capture the ordering of the sequence of activities after the execution of the first activity. Essentially, the aim of chain patterns is to define the requirements pertaining to complex pairing of individual proposition relations. The LTl formula for the *Chain Precedes* pattern is as follows:

- The *Chain – Precedes* pattern specifies that a sequence of activities σ, τ must be preceded by the occurrence of activity ϕ .

LTl Formula $(F(\sigma \wedge XF(\tau))) \rightarrow (\neg\sigma)U(\phi)$

The pattern is only useful for structural compliance checking of the processes where the norms can prescribe conditions on the ordering of the activity occurrence. However, the pattern is not suitable for representing the semantics of any obligation modality in our classification model.

Correspondence Ordering pattern only

Pattern-5 (*LeadsTo*)

Description indicates that the execution of the activity ϕ must always be followed by the execution of activity ψ .

LTL Formula $G(\phi \rightarrow F(\psi))$

Evaluation The **LeadsTo** pattern (or *Response pattern*) is a cause-effect pattern describing the relationship between two propositions ϕ and ψ . Essentially, the relationship is established when the execution of ϕ (the cause) must be followed by the execution of the ψ (the effect). Generally, the properties of the **LeadsTo** pattern frequently occurs in the concurrent systems (Dwyer et al. 1999), and have a contrary relationship with the **Precedes** pattern; in other words, they cannot be equivalent. Accordingly, the variant **Chain-LeadsTo** indicates that activity ϕ must be followed by a sequence of σ, τ ; and for a reverse interaction, the sequence of σ, τ must be followed by activity ϕ . The LTL formulas for the two cases of the **Chain-LeadsTo** pattern are respectively, as follow:

- [Case 1 ϕ LeadsTo(σ, τ): $G(\phi \rightarrow F(\sigma \wedge XF(\tau)))$
- [Case 2 (σ, τ)LeadsTo ϕ]: $G(\sigma \wedge XF(\tau) \rightarrow X(F(T \wedge F(\phi))))$

Similar to its counterpart **Precedes**, the pattern is useful only for structural compliance checking of the processes. However, given their property, they have different semantic conditions which do not correspond to any of the obligation modalities.

Correspondence Ordering pattern only

Pattern-6 (*Exists-Often*)

Description indicates that proposition ϕ must occur frequently within the process model.

LTL Formula $GF(\phi)$

Evaluation The **Exist-Often** pattern implies the frequent occurrence of proposition ϕ during the whole execution of the process. Essentially, conversely to the **Exist-Bounded** pattern, this pattern can have indefinite occurrences of the propositions. From a legal norm perspective, the pattern is not suitable for providing semantic representation of any obligation modality; however, from a structural compliance rule perspective, it can be used to model an activity that has to occur multiple times during the execution of the process. The pattern is comparable to the **isUniversal** pattern; the only difference is that the occurrence of the proposition ϕ must hold for all the instances of the portion of the process where the obligation is in force. In contrast, the **Exist-Often** proposition ϕ might appear in many of the instances of the part of the process (that is, ϕ can occurs several times in part of the process). Hence, the pattern cannot be used to represent maintenance obligation because, with the **isUniversal** pattern, it might be possible that the

obligation defining the conditions for proposition ϕ is triggered only once for the duration of the validity of the obligation. However, with this pattern, the obligation has to be triggered for every instance where the proposition needs to occur. Thus, it cannot give a full representation of the semantics of maintenance obligations. Accordingly, the pattern is dual of **Absence-Often**, and can be used to specify the negation and explicit queries for existing to define an instance of the absence pattern (Dwyer et al. 1999).

Correspondence Ordering pattern only

Pattern-7 (*DirectlyFollowedBy*)

Description shows that the required proposition ϕ is to be followed by another proposition ψ .

LTL Formula $G(\phi \rightarrow X(\psi))$

Evaluation The **DirectlyFollowedBy** pattern defines the relationship between two propositions ϕ and ψ , where if second proposition ψ occurs then the first proposition ϕ must have occurred *immediately before* ψ . The pattern is useful only for structural compliance rules, as it does not seem to have conceptual relevance to the legal norm; thus, it cannot be used to represent any obligation modality.

Correspondence Ordering pattern only

As previously argued, what makes deontic effects distinctive from other normative effects is that they can be violated. Generally, violations lead to the imposition of penalties. In some violations cases, however, corrective measures can still make the process compliant. Hence, it is particularly important that a CMF is able to handle the compensatory actions to amend the violations. More recently, in addition to the patterns discussed above new patterns have been introduced in CRL (Elgammal et al. 2014), for the specification and verification of *compensation actions* namely: *Else* and *ElseNext*. These patterns are conjunctively built with *LeadTo* and *DirectlyFollowedBy* atomic patterns as:

$$\begin{aligned} &\phi(\text{LeadsTo}|\text{DirectlyFollowedBy})\phi_1(\text{Else}|\text{ElseNext}) \\ &\phi_2 \dots (\text{Else}|\text{ElseNext})\phi_n \end{aligned} \quad (1)$$

where ϕ is the rule condition, ϕ_1 is the primary action, and ϕ_2, \dots, ϕ_n are compensatory actions. Essentially, the compensation pattern implements the *if-then-else* conditional structure of the compensatory rules. Accordingly, *DirectlyFollowedBy* and *LeadsTo* define the ordering of the primary activity, whether ϕ_1 directly occurs *immediately after* ϕ or at some time in future. The LTL equivalence formula for compensatory patterns is as follows:

$$\begin{aligned} & \mathbf{G}(\phi \rightarrow \mathbf{F}[\mathbf{X}(\phi_1 \wedge_{1 \leq i < n-1} (\mathbf{F}[\mathbf{X}(\phi_i \text{NotSucceed}) \wedge \\ & (\phi_i \text{NotSucceed} \rightarrow \mathbf{F}[\mathbf{X}(\phi_{i+1})])])))) \end{aligned} \quad (2)$$

ϕ gives the antecedent of the compensatory rule, that is, the rule's conditions; ϕ_1 is the head of the rule representing the primary action that must be taken; ϕ_2, \dots, ϕ_n represents the compensatory actions that must be taken if the conditions of the rule are violated; and i is a natural number; that is, $n \in \mathbb{N}$; and $\phi_i \text{NotSucceed}$ represents the decision point that checks whether ϕ_i holds.

In addition to the generic rules patterns, CRL offers patterns for modeling non-monotonic requirements. This allows CRL to model *exceptions*. More specifically, exceptions provide conditions under which the primary requirement might not hold. Following Baral and Zhao (2007), CRL has two patterns for exceptions: one for *strong exceptions* and one for *weak exceptions*: A strong exception on the primary rule mandates that whenever the strong exception holds, the primary rule *must* not hold. A weak exception, on the other hand, indicates that when the weak exception holds, the primary rule *might* or *might not* hold. The patterns for strong and weak exception are as follows:¹¹

1. *strong exception*: $[[R]]\text{pattern}$ and
2. *weak exception*: $[R]\text{pattern}$

where $[[R]]$ and $[R]$ are the LTL formulas encoding the exception conditions and *Pattern* is the LTL formula corresponding to the primary requirement the exception applies to.

Given the potential recursive nature of exceptions (that is, having the ability to capture exceptions to exceptions), CRL recursively resolves the dependencies of the exception conditions, using the following translation to LTL

1. $[[R]]\text{pattern}$ is translated to $\phi \rightarrow \neg\psi$,
2. $[R]\text{pattern}$ is translated to $\phi \vee \psi$;

where ϕ is the LTL formula corresponding to R and ψ is the LTL counterpart of *Pattern*.

The above analysis of the COMPAS patterns shows that majority of the patterns are ordering patterns only; however, patterns 4–7 can be used to determine what and when obligations are in force. Currently, the framework is only able to represent a fraction of obligation modalities, and the support for all types of normative requirements is limited. This is because the use of LTL as its underlying formal language, which has limited abilities to give a faithful representation of legal norms as formally proved by Governatori and Hashmi (2015).

¹¹ The brackets determine the relative strength of the exception, where $[[\]]$ indicates *strong exception* and $[\]$ represents *weak exception* respectively.

3.3 DECLARE

Declare (Pesic and van der Aalst 2006) is a compliance management system based on a rather flexible approach for run-time verification of declarative process models. Contrary to imperative models which specify rigid conditions on how the model should work, declarative models are flexible in specifying what (minimal) set of constraints must be fulfilled for a successful business interaction. The business knowledge in the Declare models is represented by means of business constraints using Constraint Declarative¹² [ConDec (Pesic and van der Aalst 2006)] language. ConDec provides graphical notations for representing the flow of business interaction as well as several control-independent abstractions for constraining activities in the model. The system includes two types of constraints, that is, mandatory and optional to design the constraint models. Mandatory constraints are ones which cannot be violated, whereas optional constraints allow to follow a preferable course of actions but can be deviated from them. Declare incorporates two different tools namely: YAWL and ProM¹³ for execution and verification of the Declare models respectively. The Declare model is executed by the YAWL workflow engine that is used to check for any non-complaint interaction between the tasks of the process. A process can only be active if it does not violate a mandatory constraint, and the model can be fully executed only if all the constraints are fulfilled. The compliance of the constraints is reported as *satisfied*, *temporarily violated* and *violated*. If the model satisfies all the constraints, the interaction is completed; however, if there is a violation state, the process is not allowed to satisfy the constraint. For temporarily violated states; however, the Declare system allows to satisfy the constraint.

The business knowledge in the Declare framework are modelled in the form of *Declare expressions*, using ConDec language. These expressions are grouped into four categories, namely: *existence*, *relations*, *choice* and *negative* constraints. Table 4 illustrates the Declare constraints and their meanings. Mostly, these constraints represent obligations, prohibitions constraints and correspond to LTL formulas that provide the semantics for Declare expressions.

We now examine in details the Declare's constraints in each group to determine whether they have correspondence with the obligation modalities.

Existence Constraints Declare provides four generic patterns, and one special unary constraint pattern that define the number of times an activity might or might not occur in a trace or during the whole execution of the process as illustrated in Table 4. Essentially, existence patterns have visual resemblance to UML's multiplicity constraints (Pesic et al. 2007). The existence patterns are categorised into: $\text{Existence}(n, \phi)$, $\text{Existence}(n, \phi)$, and two absence patterns $\text{Absence}(n+1, \phi)$, $\text{Absence}(\phi)$. The aim of the existence constraints is to specify that the activity ϕ must be present in the execution of trace, whereas absence constraints specify that activity ϕ must occur in the trace. The special $\text{init}(\phi)$ cardinality pattern has

¹² In Nov 2012, the name of the ConDec language was changed to Declare, see <http://www.win.tue.nl/declare/2011/11/declare-renaming/>.

¹³ ProM Tools: <http://www.promtools.org/doku.php>.

Table 4 Declare constraint patterns and meanings (Pesic and van der Aalst 2006)

Constraint name	Meanings
<i>Existence constraints</i>	
Absence(ϕ)	Activity ϕ cannot be executed
Absence($n+1, \phi$)	Activity ϕ can be executed <i>at most</i> n times
Existence(n, ϕ)	Activity ϕ must be executed <i>at least</i> n times
Existence(n, ϕ)	Activity ϕ must be executed <i>exactly</i> n times
Init(ϕ)	Activity ϕ must be the first executed activity
<i>Relation constraints</i>	
Resp_existence($[\phi], [\psi]$)	if ϕ is executed, then ψ must be executed <i>before or after</i>
Coexistence($[\phi], [\psi]$)	<i>neither</i> ϕ nor ψ is executed, or they <i>both</i> are executed
Response($[\phi], [\psi]$)	if ϕ is executed, then ψ must be executed <i>thereafter</i>
Precedence($[\phi], [\psi]$)	ψ can be executed only if ϕ has been <i>previously</i> executed.
Succession($[\phi], [\psi]$)	ϕ and ψ must be executed in <i>succession</i> ; i.e., ψ must follow ϕ and ϕ must precede ψ
Chain_response($[\phi], [\psi]$)	if ϕ is executed, then ψ must be executed <i>next</i>
Chain_precedence($[\phi], [\psi]$)	if ψ is executed, then ϕ must have been executed <i>immediately before</i> ψ
Chain_succession($[\phi], [\psi]$)	ϕ and ψ must be executed in <i>sequence</i>
Alt_response($[\phi], [\psi]$)	ψ is <i>response</i> of ϕ , and <i>between</i> every two executions of ψ , ϕ must be executed at least once
Alt_precedence($[\phi], [\psi]$)	ϕ is <i>precedence</i> of ψ , and <i>between</i> every two executions of ψ , ϕ must be executed at least once
Alt_succession($[\phi], [\psi]$)	ψ is <i>alternate response</i> of ϕ , and ϕ is <i>alternate precedence</i> of ψ
<i>Negation constraints</i>	
resp_absence($[\phi], [\psi]$)	if ϕ is executed, then ψ can <i>never</i> be executed
Not_Coexistence($[\phi], [\psi]$)	ϕ and ψ <i>exclude</i> each other
Neg_response($[\phi], [\psi]$)	ψ cannot be executed <i>after</i> ϕ
Neg_precedence($[\phi], [\psi]$)	ϕ cannot be executed <i>before</i> ψ
Neg_succession($[\phi], [\psi]$)	ϕ and ψ cannot be executed in <i>succession</i>
Neg_alt_response($[\phi], [\psi]$)	ψ cannot be executed between two occurrences of ϕ
Neg_alt_precedence($[\phi], [\psi]$)	ϕ cannot be executed between any two ψ s
Neg_alt_succession($[\phi], [\psi]$)	ψ cannot be executed between any two ϕ s and vice-versa
Neg_chain_response($[\phi], [\psi]$)	ψ cannot be executed <i>next</i> to ϕ
Neg_chain_precedence($[\phi], [\psi]$)	ϕ cannot be executed <i>immediately before</i> ψ
Neg_chain_succession($[\phi], [\psi]$)	ϕ and ψ cannot be executed in <i>sequence</i>

different meanings, as it specifies that the ϕ must be the first executed activity in the model.

Evaluation The Declare existence constraints have similar meanings to the COMPAS atomic patterns `isAbsent`, `Exists` and `Bounded Exists`. Hence, these patterns will have the same expressiveness to represent obligations and prohibitions.

The only difference is that Declare's patterns explicitly define the cardinality of the constraints; that is, the number of occurrences.

Correspondence The existence patterns can support *achievement* obligations and *prohibitions*.

Relation and Negative Constraints Relation constraints are *binary* constraints that define positive dependency between two activities in a trace. In other words, these constraints impose the conditions that—the presence of an activity ϕ is bounded with the presence of another activity ψ ; thus, the relation constraints are reactive. Essentially, most of the relation constraints specify the execution order of activities; thus, possibly impose qualitative temporal constraints between activities (Pesic and van der Aalst 2006). If the temporal ordering is not followed, it results in non-compliance of the constraint. However, **Resp_existence** and **Coexists** are relation constraints that do not specify any temporal ordering, thus leaving the activity to execute freely. For example, if activity ϕ occurs, then the activity ψ must also occur in the trace; however, the temporal ordering does not matter if ϕ occurs first or after of ψ .

The **response**, **precedence**, **succession**, and extended variant of **resp_existence** impose strict temporal ordering between activities. **Alt** and **chain** constraints, on the other hand, impose even tighter conditions on the ordering of the execution between activities, thus making **response**, **precedence** and **succession** even stronger. Essentially, relation constraints allow defining different positive relations between the activities, from very loose to very strict temporal ordering relations. Table 4 depicts the relation constraints.

In addition to relation constraint, Declare also offers negated variants of relation constraints. Negative constraints aim to prevent the occurrence of activity(-ies) between the time bounds defined by some other constraint (Pesic et al. 2007). In other words, a negative constraint means that some activity must be absent (must not occur) until the occurrence of some other activity defining a negative relation between them; that is, activities are incompatible. However, it must be noted that negative relation would not mean logical negation of the constraint; rather, both negative and positive constraint can be interpreted as true during the execution of the process. For example, if **Resp_existence** is true then its negation **Neg_Resp_existence** can be evaluated as true in the trace. Table 4 illustrates the negative variants that Declare framework offer for all relation constraints. Essentially, negative constraints can be used to specify the constraint prohibiting some actions.

Evaluation Most of the Declare's relation constraints have similar expressive power to the COMPAS's atomic and composite patterns, because these patterns are based on LTL. In addition, these constraint patterns also have similarities with BPMN-Q patterns, which are based on CTL. Since CTL is the superset of LTL, these patterns in these frameworks share the same limitations when it comes to representing different types of obligations. Thus, relation constraints are suitable only for modeling structural compliance rules imposing temporal ordering

constraints. However, negative relation constraints can be used for representing *prohibition*.

Correspondence Negative constraint can have correspondence with *prohibitions*, but no correspondence with relation constraints as they are ordering constraints only.

From the above evaluation it can be seen that Declare has a very limited scope, as currently it can only model achievement obligations and prohibitions, whereas other types of norms cannot be explicitly modelled (see Table 9). The representation of achievement obligations is only possible because such obligations prescribe deadlines, and the contents of the obligation must be achieved at least once. Declare models with such constraints prescribing deadlines can be supported because the tasks of the obligation will be carried out in some future time. However, other types of obligations—for example, *persistent* and *preemptive* obligations cannot be modelled. Accordingly, representing constraints prescribing *maintenance* obligations is not possible because the obligation conditions must be true for the whole duration of the process. However, it might be possible that there are some situations when the obligation contents might be present, which results in a deadlock between the tasks of the process. Moreover, Declare system is not able to handle violations because of the non-deterministic behaviour of declare models and lack of declarative nature of linear temporal logic (LTL). Hence, if a violation of the constraint is detected in the model—further processing will be stopped. Accordingly, due to the issues with LTL representing permissions, compensation and perdurant obligations is not possible as formally proved in (Governatori 2015).

Remark 1 Declare also provides choice (branching) constraints that are concerned with a multiplicity of activities (see Pesic et al. 2007, for more details). Choice constraints are interpreted as disjunctive, and aim to combine reactive behaviour of relation and negative constraints. However, they still have the same limitations as relation constraints; hence, we safely omit them from our evaluation.

3.4 Business process modelling notations-query language (BPMN-Q)

Business Process Modelling Notation-Query (BPMN-Q, Awad et al. (2008a, 2011)) is a query-based visual language for structural compliance checking of process models. The language uses visual patterns to model various types of compliance rules e.g., control-flow, data and conditional rules etc., which are translated into CTL expressions. The compliance checking of Computation Tree Logic (CTL) expressions is performed using a systematic approach where first the query templates representing the patterns of compliance rules are generated. These templates are then used to identify the process models called sub-graphs from the process repository subject to compliance checking. The extracted query templates are structurally matched to the extracted sub-graphs using a query processor. Once the sub-graphs are extracted they are reduced by removing the elements not required for the interaction, and then transformed into Petri Nets models. CTL formulas are then generated from the translated Petri Net models, which are then inserted into a

Table 5 BPMN-Q constraints name, CTL representation and meanings (Awad et al. 2011)

Constraint name	CTL representation	Meanings
Global – scopepresence (ϕ)	$AG(start \rightarrow AF(executed(\phi)))$	Aims to reflect the requirements of executing a certain activity ϕ in all the instances
Global – scopeabsence (ϕ)	$AG(start \rightarrow A[\neg executed(\phi)U end]))$	Indicates the requirement of not executing an activity ϕ by assuring that there is no state of the process execution, from start to end, where ϕ was executed
Before – scopepresence (ϕ, ψ)	$\neg E[\neg ready(\phi)U ready(\psi)]$	Also known as precedence indicating that the execution of activity ψ is preceded by the execution of another activity ϕ
Before – scopeabsence (ϕ, ψ)	$\neg EF(start \wedge EF(executed(\phi) \wedge EF(ready(\psi))))$	An activity ϕ might be required to be absent before the execution of another activity ψ
After – scopepresence (ϕ, ψ)	$AG(executed(\phi) \rightarrow AF(executed(\psi)))$	Also known as response pattern, indicates that after the execution of an activity ϕ , another activity ψ has to be eventually executed
After – scopeabsence (ϕ, ψ)	$AG(executed(\phi) \rightarrow A[\neg executed(\psi)U end]))$	Similar to before-scope absence, the pattern indicates that certain activities ϕ are prohibited to be executed after the execution of another activity ψ
Between – scopeabsence (ϕ, ψ, φ)		Indicates that between the execution of two activities ϕ and φ , it is prohibited to execute certain other activities ψ . There are two variations of <i>Between-scope absence</i> namely
1. $AG(executed(\phi) \rightarrow A[\neg executed(\psi)U executed(\varphi)])$		
[<i>Response with Absence</i>] indicates activity φ is required to be executed after ϕ while there is no chance in between to execute ψ		
2. $\neg E[\neg executed(\phi)U ready(\varphi)] \wedge \neg EF(executed(\phi) \wedge EF(executed(\psi) \wedge ready(\varphi)))$		
[<i>Response with Absence</i>] indicates whenever φ is executed; ϕ must have been executed before without executing ψ in between		

model checker. The model checker processes the CTL formulas to detect any violations. The compliance results are reported in the form of YES/No type answers, where YES indicates that the process is compliant, and No indicates that the process does not comply with the query templates.

BPMN-Q uses visual notations to model different types of rules e.g., control-flow, data and conditional rules as patterns based on different occurrence Scope property specification patterns, as proposed in Dwyer et al. (1999), depicted in Table 5. The *scope patterns*, a special case of occurrence patterns, aim to define the ranges (called *regions*) over which the pattern must be evaluated as true. With the scope patterns, the requirements pertaining to the existence or absence of certain propositions over the ranges of process executions are defined. The range of the scope is determined by defining the start and end point of the range. Five scope patterns can be derived, namely: *global*, *before*, *after*, *afteruntil*, and *between scope*. The *global pattern* defines the range over the whole execution of the processes, whereas *before pattern* specifies the ranges before which the proposition must be true. Meanwhile, *after scope* defines the range from which the proposition must hold true. In contrast, *between scope* specifies the ranges at any portion of the process execution in which the proposition must hold true. Similar to between, the *after-until pattern* specify the range until which the proposition must continue to hold, even if the until part of the range is not executed.

BPMN-Q uses the scope patterns to express different types of obligations by means of visual patterns, as illustrated in Fig. 3.¹⁴ Each of these patterns, similar to the standard BPMN notations, contains a CTL formula giving the specifications of a specific obligation type. For example, the pattern Fig. 3a shows the global-scope pattern referring to a single activity that might be required to be executed in all process instances. Maintenance obligations can be expressed using *global space presence* patterns (shown above), which enable the execution of specific activity throughout the process. Conversely, the pattern Fig. 3c illustrates the case where a certain activity must not execute at all. Since prohibitions must be observed in any case during the execution of a process they are represented by *global space absence* patterns.

Accordingly, BPMN-Q also provides constructs to model obligation types where the obligations might prescribe the conditions applicable to the data and resources aspects of a business process. BPMN-Q incorporates most of the patterns used in COMPAS; thus, the framework is able to handle almost the same type of obligation modalities as COMPAS (cf. Table 9). On the same note, BPMN-Q does not provide any conceptual or formal constructs that can be used to model permissions. Similar to COMPAS, BPMN-Q is also able to handle the violation of obligations. A graphical violation-handling approach using *anti-patterns* is discussed in (Awad and Weske 2009). The anti-patterns are derived from the BPMN-Q visual patterns, and are comparable to *LeadsTo* and *isAbsent* patterns (see Awad and Weske (2009) for details on anti-patterns). Finally, similar to Declare framework, BPMN-Q's visual patterns are not suitable to model the specifications of the cases of permissions, compensations and perdurant obligations (Governatori 2015). This is

¹⁴ This is not an exhaustive list of the BPMN-Q patterns, see (Awad et al. 2011) for a detailed list.

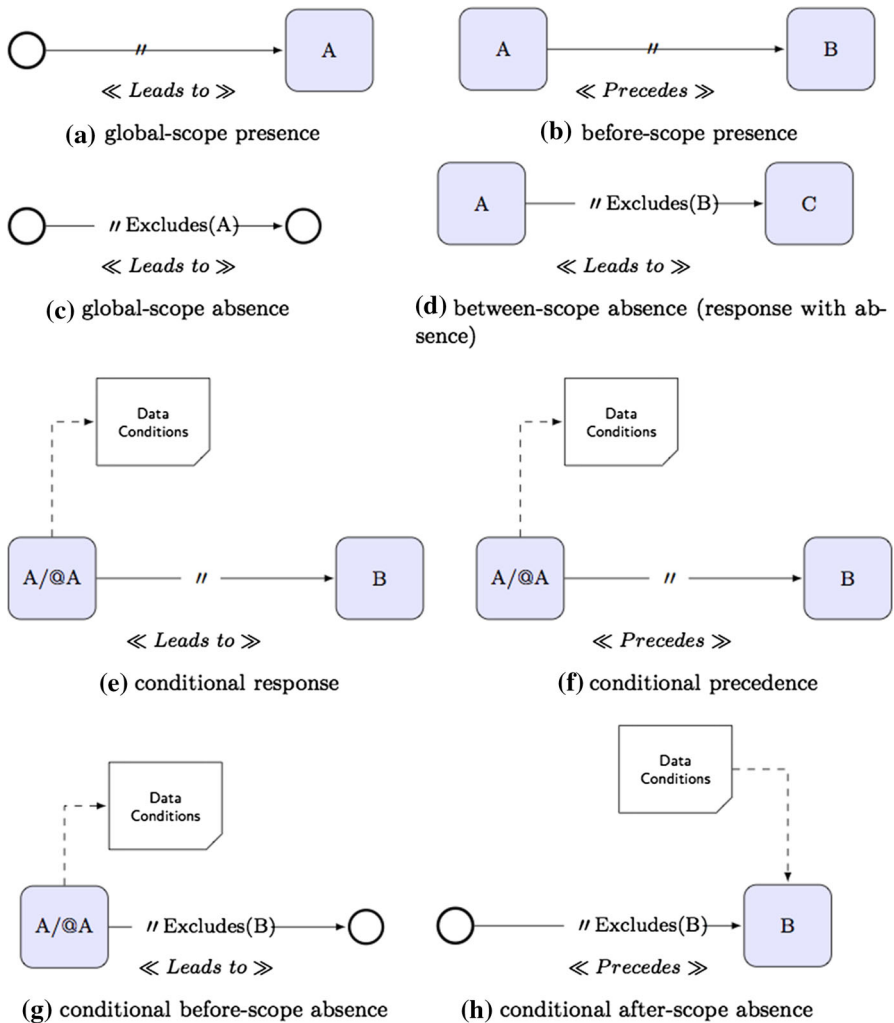


Fig. 3 List of BPMN-Q visual patterns to model norms Awad et al. (2011)

because BPMN-Q is based on CTL which is superset of LTL, thus the issues with LTL for representing normative requirements can be generalised to CTL based frameworks as well.

3.5 SEAFLOWS

SeaFlows (Ly et al. 2010b, 2012) is a compliance-by-design framework for behavioural and structure compliance checking of the structured process models. It uses a primitive-based graphical language to model complex compliance rules. The compliance rules are represented using *first-order-logic* (FOL) predicates instanciable to the compliance rule graphs. SeaFlow proposes a structural compliance

checking approach to verify the compliance of applicable rules whether the process model violates the derived structural criteria called, templates to detect any non-compliant patterns. Compliance checking is carried out in three steps where in the *first* step, a set of process models based on the nodes relations is automatically derived, which is then compared with the derived templates to detect any anomalies in the *second* step. *Finally*, all the queried templates are combined and fed into the compliance checking module. A ‘YES’ is reported if no non-compliant behaviour is detected i.e., the process model fully complies with all the instantiated rules, a ‘NO’ is returned otherwise.

SeaFlow uses a compositional graph-based approach to represent antecedent consequence type rules in the form of compliance rules graphs (CRGs) Ly et al. (2010b). The CRGs are place holders for FOL expressions representing compliance rules. SeaFlow provides four CRGs, each indicating occurrence and absence of activities of an associated type namely: ANTEOcc and ANTEAbs which are used to model the antecedent pattern triggering a compliance rule. CONOcc and CONAbs, on the other hand, map the consequence patterns.

In addition, the ordering of the nodes in a CRG is defined using a relation primitive, whereas a data conditions primitive is used to represent the data conditions of a compliance rule. The CRGs are not merely visual notations; they are also equipped with the formal semantics for checking verification of the process models. This is because SeaFlows is able to check the compliance of behavioural as well as structural compliance rules.

The framework defines five structural patterns as criteria for determining the compliance status of the process. These criteria can be considered as queries on the relations of the nodes of the process model. The structural criteria patterns are: *containment*, *occurrence*, *precedence*, and *precedence relation* as shown in Table 6.

Next, we examine these patterns in detail to check their correspondence with the obligation modalities.

Pattern-1 (Contains (ϕ))

Table 6 SeaFlows’ structural criteria adopted from Ly et al. (2010b)

Structural Criteria	Usage
Contains (ϕ)	A unary structural containment relation that indicates if ϕ is contained in the process model
(ϕ) Excludes (ψ)	A structural occurrence relation that indicating whether ϕ and ψ are located on different branches of an exclusive gateway
(ϕ) Implies (ψ)	A non-directed structural occurrence relation that indicates ψ must not be located on the same branch of an exclusive gateway, on which ψ is located, such that ϕ and ψ both exist
(ϕ) Implies ($\psi_1 \psi_2 \dots \psi_n$)	A non-directed structural occurrence relation indicating that whether A is always executed together with ψ_1, ψ_2, \dots or ψ_n
(ϕ) Precedes (ψ)	A structural precedence relation that indicates if there is a directed path in the process model leading from ϕ and ψ

Syntax $\oplus \phi$

Description The unary structural contains relation whether ϕ is in the process model, and \oplus is the **contains** operator.

Evaluation The **Contains** ϕ pattern (a.k.a *eventually*) is used to indicate if the proposition ϕ is contained in the portion of the process model. The pattern is similar to the COMPAS and Declare **Existence** pattern, and comparable to CRL's **Bounded Existence** where one may specify the number of occurrences at most (or at least) some bounded number of times. The most common example of **Contains** is specifying termination; for example, on all the executions of the process model eventually, we reach a terminating state (Dwyer et al. 1999). As far as using the **Contains** patterns is concerned **Contains** ϕ can be used to represent achievement obligations because by the persistent obligations definition, the obligation will hold in some future time.

Correspondence The pattern can be used to represent *achievement* obligations.

Pattern-2 $((\phi) \text{ Excludes } (\psi))$

Syntax $\phi \otimes \psi$

Description In the process model, the structural relation patterns indicates that whether ϕ and ψ are located on different branches of an exclusive gateway, and \otimes is an **excludes** relationship operator.

Evaluation The **Excludes** pattern defines the relation between two activities, indicating that ϕ and ψ do not exist in the same trace of the process model. This contra relation pattern is similar to COMPAS's **Exclusive**, and Declare's **Not-CoExistence** pattern that specifies that two activities are incompatible (Montali 2010). For example, a rule might put constraint that if one activity (ϕ) occurs, then the other activity (ψ) cannot occur at the same time; for example, if an order is accepted, it cannot be rejected in an interaction with the customer. Although the pattern is useful for representing structural compliance rules, similar to BPMN-Q's **Global-Scope-Absence** visual pattern, it might be useful for representing prohibitions.

Correspondence The pattern might be useful to model *prohibitions*.

Pattern-3 $((\phi) \text{ Implies } (\psi))$

Syntax $\phi \triangleright \psi$

Description The structural ordering pattern defines the relation between two activities ϕ and ψ , prescribing the condition that both activities must not be in the same execution trace; however, they must occur in the execution of the whole process. \triangleright is a **implies** operator.

Evaluation Essentially, the pattern is suitable for modeling structural rules that stipulate contra relation conditions for two activities. For example, if the order is less than \$50,000, then no solvency check is required if the customer has a premium

status. A composite expression ($\text{CoExists} \wedge \neg \text{CoAbsent}$), built with COMPAS's atomic patterns, can give the similar meanings that a solvency check cannot co-exist with the customer's premium status on the same branch of the exclusive gateway. As the pattern defines a negative relation between two activities, it can be used to represent prohibition-based deontic norms because prohibitions do not specify temporal properties.

Correspondence The pattern can be used to model *prohibitions*.

Pattern-4 $((\phi) \text{ Implies } (\psi_1 | \psi_2 | \dots | \psi_n))$

Syntax $\phi \triangleright \psi_1 | \psi_2 | \dots | \psi_n$

Description The pattern shows a non-directed structural occurrence relation, indicating whether ϕ is always executed together with $\psi_1, \psi_2, \dots, \psi_n$.

Evaluation The **Implies** pattern defines the occurrence relation between an activity ϕ and a set of activities $\psi_1, \psi_2, \dots, \psi_n$, where the activities are co-located in a set of execution trace. A Declare expression giving similar meanings can be built by combining **resp_existence** and **Coexistence** patterns, as the **Implies** pattern properties do not specify any ordering of the activity execution. However, the pattern can be used only for checking the compliance of structural rules, and cannot be used for representing the temporal properties of any obligation modality of legal norms.

Correspondence –NA– (ordering pattern only)

Pattern-5 $((\phi) \text{ Precedes } (\psi))$

Syntax $\phi \gg \psi$

Description The **Precedes** precedence relations indicates whether there is a directed path in the process model leading from ϕ and ψ . The **precedes** operator \gg denotes the precedence of the relationship between two propositions.

Evaluation The pattern has similar objectives to the **Precedes** pattern of COMPAS and Declare as both frameworks also provide as extended variant **Chain-Precedes** to capture the ordering sequence of activities after the execution of the first activity. SeaFlows does not offer the extended **Chain-Precedes** variant; however, the **Chain-Precedes** is also extensible to SealFows. As far as modeling obligation modalities is concerned, the **Precedes** pattern has the same limitations as COMPAS framework.

Correspondence –NA– (ordering pattern only)

As it is clear (from above), SeaFlows is able to model achievement obligations that stipulate the occurrence of some event in the future by means of the occurrence pattern $(\oplus \phi)$. Essentially, the SeaFlows patterns are useful from a structural compliance of business processes perspective only. These patterns have no temporal relevance to the semantics of our obligation modalities. This is because they are based on **ANTEOcc** and **CONSOcc** primitives of the CRGs. The CRGs are based

on FOL formulas, and provide the formal semantics of structural compliance rules. However, with the formulas representing the CRGs, it is not possible to give a one-to-one mapping of temporal semantics of other obligation modalities such as permissions, prohibitions, compensation, and maintenance (see Table 9). This is because first-order-logic (FOL) is not able to distinguish between expressions corresponding to factual statements (something is true or false) from expressions establishing that something is obligatory (or forbidden or permitted). Obligations and other deontic modalities can be represented by higher-order predicates. As Herrestad (1991) pointed that this FOL (or better higher-order-logic) representation still suffers from some problem, in particular in presence of norms that can be violated and admit compensation; in some situations, we may get redundant information especially where rules are violated, and depending on the conditions new rules get the effects that might convey new information which is not available in other propositions of the rules. Hence, these issues restricting the effectiveness of FOL making it unsuitable for providing reasoning support for all types of normative requirements. However, FOL has the ability to evaluate the *truth-functional* values of the domain formulas representing the legal norms over the possible executions of processes. The only possibility to check the compliant status of the norms is by means of model-checking, provided there is an external oracle performing the legal reasoning to determine what obligations are in force and when they are in force. Essentially, this is the same limitation for the approaches that are based on temporal logic as illustrated by (Governatori 2015; Governatori and Hashmi 2015).

3.6 Process compliance language (PCL)

Process compliance language (PCL Governatori and Rotolo 2010) is a formal system that provides strong conceptual foundations to provide reasoning and modeling support for legal norms and other aspects of legal reasoning. PCL is based on Defeasible Logic (Antonioni et al. 2001), an efficient non-monotonic logic, and a deontic logic of violations (Governatori and Rotolo 2006), which allows the formalism to deal with exceptions and effectively handle violations and contrary-to-duty obligations. The language uses a set of propositional literal to represent the state variable and tasks in a process. Moreover, it provides logical operators such as negation \neg , a non-Boolean connective \otimes , and other deontic operators to model different obligations modalities, which are used to write the PCL formulas called *PCL specifications*. For compliance checking, the PCL specifications representing the norms are annotated to the tasks of the process. These specifications are either automatically extracted from the databases or other sources attached to the processes (Hashmi et al. 2012) or they are directly provided by business analysts. The annotated processes are then assessed for any non-compliance issues. PCL uses a three-step algorithm, which first computes the set of all possible effects for the tasks of the process. Then, it determines which obligations are in force for each task. Finally, all active obligations are compared with the effects of the task to determine whether the process deviates from its intended behaviour. The status report is returned either as *fully compliant*, *partially compliant* or *not compliant*.

The use of deontic and defeasible logic enables PCL to provide conceptually sound reasoning and modeling support for all classes of obligations (see Table 9) and other aspects—for example, reasoning with the superiority relation of compliance rules, and reasoning with the contrary-to-duty norms, to name but a few. This is because *deontic logic* effectively deals with the modeling and handling of violations of obligations and chains of reparations. *defeasible logic*, on the other hand, tackles the problem of partial information and anomalous prescriptions. PCL provides three main operators for basic deontic notions: punctual (O^p), maintenance (O^m), and achievement (O^a). Depending on the conditions, *achievement obligations* can be further classified into *preemptive* and *non-preemptive*, and *perdurant* and *non-perdurant obligations* (Hashmi et al. 2013). PCL allows to explicitly define the deadlines in the formalised PCL expressions, where it allows the algorithm to distinguish between different types of obligations. Table 7 illustrates various deontic operators in PCL to model different types of obligations.

Effective management of violations is one of the major requirements for a CMFs. This is because—often times processes operate in highly unpredictable environments. Due to some external circumstances, they might deviate from the prescribed behaviour, thus result in the violations. PCL offers effective management of the situations where obligations are violated and then compensated to ensure compliance. For handling violations, PCL provides a special contrary-to-duty \otimes operator to create reparation chains to manage multiple violations. As far as the persistence of obligation after the violation is concerned, the notion of perdurant obligation has not been addressed in the current version of PCL however, the notion has been addressed elsewhere (Allaire and Governatori 2014), enabling PCL to offer a holistic and more conceptually rich and sound reasoning support for all classes of norms.

3.7 Business process compliance auditing framework

The business process compliance auditing framework (Ghose and Koliadis 2007) is an auditing system that proposed a heuristic approach to verify the compliant status of business processes against the regulatory rules. The framework is able to identify and detect a range of compliance issues, and provide automated compliance resolution support for effective decision support.

Table 7 Types of obligations operators in PCL (Governatori and Rotolo 2010)

Obligation operators	Meanings
O^p	Punctual
$O^{a,\pi}_{pr}$	Achievement, persistent, preemptive
$O^{a,\pi}_{n-pr}$	Achievement, persistent, non-preemptive
$O^{a,\tau}_{pr}$	Achievement, non-persistent, preemptive
$O^{a,\tau}_{n-pr}$	Achievement, non-persistent, non-preemptive
O^m	Maintenance

To verify the compliance, the analyst first accumulates all the effects related the process models. This gives the description of local context because the compliance checking is performed locally, where the accumulated rules are relevant to the part of the process. Once all the relevant effects are accumulated, the process model is annotated with these effects. After that, directed graphs called *Semantic Process Networks* (SPNs) (Ghose and Koliadis 2007) translating the annotation processes are derived. These SPNs are then used to validate the compliance of the process using an algorithm that traverses the traces of the effects annotated processes for any non-compliant issues, and returns the compliance status report. If a violation is detected, the analyst must modify the process model and recheck it for compliant execution.

The compliance rules in this CMF are attached by means of parsimonious effect annotations (Ghose and Koliadis 2007; Hinge et al. 2009). There are two types of effect annotations that can be derived from the literature *formal and informal*. The said framework incorporates formal annotation effects, which are represented and parsed using CTL, a state-based logic. The parsimonious annotations are used to validate the compliant behaviour of the business processes. Unlike PCL's semantic annotations, the formal annotations in this framework cannot make any distinction between different types of obligations, as it is not clear that how such distinction can be made. As the violation of obligation largely depends on the temporal conditions (that is, deadlines), it is not possible to analyse when an obligation is violated; it is only possible to determine whether an activity annotated with the formalised rule description exists or is absent from the graph. In addition, as the framework used a heuristics based approach for asserting and resolving compliance issues, it uses the structural compliance patterns and semantic patterns. The structural patterns used in the framework are: (a) *activity/event/decision inclusion*; (b) *activity/event/decision coordination*; (c) *activity/event/decision assignment*; (d) *actor/resource inclusion*; and (e) *actor/resource interaction* (Ghose and Koliadis 2007).

Evaluation The structural patterns formalised in CTL provide the basis for resolving the non-compliance issues in the processes, albeit in a semi-automated way. Informally, the structural patterns can contain the information on the compliance rule that is violated, and on the actions to repair the problem. In contrast, the semantic patterns might contain suggestions on the required changes to amend the violation in order to restore the compliance issues. The framework proposes three semantic patterns namely: (i) *effect inclusion*; (ii) *effect coordination*; and (iii) *activity modification*.

Correspondence The auditing framework is only able to represent *achievement obligations*, this is because it uses the CTL as formal language to represent the compliance requirements; thus, has the same limitations as other LTL based CMFs. Moreover, the framework does not provide any conceptual or formal constructs to represent prohibitions, compensation and other classes of norms.

Table 8 Summary of conceptual evaluation of features of existing CMFs

CMF/approach language/system	Compliance checking approach	Modelling constructs	Underlying formalism	Linking norms	Level of support
PENELOPE	Design-Time	Norms property specifications	EC	BPMN Models are generated from deontic assignments	Partial
COMPAS	Run-Time (Structural Compliance Checking)	Domain-specific patterns	LTL	By means of CRL property patterns	Partial
DECLARE	Run-Time (Structural Compliance Checking)	Control flow-based declare expressions	Temporal Logic	Constraints specified by means of Declare expressions	Partial
BPMN-Q	Model Checking	BPMN query templates	PLTL/CTL	By means of visual patterns	Partial
PCL/FCL	Design-Time	Deontic constructs specifications	Defeasible and Deontic Logic	Specification of deontic aspects as annotations	Full
SEAFLOW	Design-Time (Structural Compliance Checking)	Compositions graph-based modeling	First order Logic	Annotations	Partial
Auditing Framework	Post Execution	Structural Patterns	CTL	By means of Parsimonious effects annotations	Partial

4 Discussion

For a CMF to be effective, it needs to be based on sound conceptual and formal models. If a CMF is not based on strong foundations, the compliance checking approach proposed in the CMF cannot be relied upon, and the compliance reporting results produced by the CMF might not be acceptable to the accrediting authorities. We evaluated the conceptual models of seven CMFs based on a pre-defined criteria—in particular, based on the different classes of normative requirements.

We examined the salient features of the selected CMFs, such as their compliance checking approach, the norms modeling constructs, their underlying formal language, and how they link normative requirements with process models. Table 8 summarises the evaluation of various features of existing CMFs. We also investigated *what is lacking* in terms of technical support in the business process compliance domain from the perspective of representing legal norms. The evaluated CMFs incorporate various strategies for checking the compliance of normative requirements, using different graph-based compliance requirements patterns, formalised using variants of temporal logic such as LTL and CTL. In contrast, some CMFs use deontic constructs and norm properties specifications such as PCL and PENELOPE, respectively.

The evaluated CMFs use different techniques *to link normative requirements* with business processes. For example, COMPAS links formalised compliance rules by means of CRL property patterns, while they are specified in terms of declarative expression in Declare. Accordingly, PCL and SeaFlows annotate business processes with compliance requirements. We also examined *the level of compliance management support for different classes of norms*. The results highlight that most of the existing CMFs provide only, a partial reasoning and modeling support for normative requirements; in other words, not all types of norms are supported in these CMFs. The exception is PCL, which offers a full modeling support for all classes of our classification model.

In the conceptual evaluation, we evaluated the conceptual foundations of the CMFs to examine which modeling constructs are provided to model different types of obligations; in particular, the constructs that can provide the reasoning and modelling support for obligation modalities. The results of the reported evaluation are summarised in Table 9, which illustrates the available support for a specific type of norms. The ‘+’ symbol indicates that the CMF is able to provide the reasoning and modeling support for a specific obligation modality, and ‘−’ indicates that the obligation modality is not supported—or it is not considered in the evaluated CMF.

From the evaluation results, it can be seen that a very limited support is available for normative requirements. Most of the CMFs fail to provide a full reasoning and modeling support for all types of norms. For example, PENELOPE can only represent obligations and permissions; no other types of norms can be represented, as EC is not suitable to model and reason about legal constraints. This is because EC lacks the modeling constructs requirements to capture the various nuances of the norms, in particular it fails to capture the effects of norms on business processes as formally shown in (Hashmi et al. 2014). Contrastingly, PCL is able to represent and

Table 9 Summary of norms support in existing CMFs

Framework	Obligations Modalities									
	Achievement	Preemptive	Non-Preemptive	Maintenance	Punctual	Perdurant	Compensation	Permission	Prohibition	Violation
PENELOPE	+	-	-	-	-	-	-	+	-	-
PCL	+	+	+	+	+	+	+	+	+	+
DECLARE	+	-	-	-	-	-	-	-	+	-
BPMN-Q	+	-	-	-	-	-	-	-	+	+
SEAFLOWS	+	-	-	-	-	-	-	-	+	+
COMPAS	+	-	-	+	-	-	+	-	+	+
AUDITING BPC	+	-	-	-	-	-	-	-	-	-

reason about all obligation modalities, thanks to the deontic and non-monotonic properties as the core of the formal logics it uses. In addition to representing basic deontic notions, PCL is also able to effectively handle violations and contrary to duty obligations—especially, time varying obligations such as achievement obligations, their properties and persistence effects overtime. However, the language of PCL is restricted to literals.

Since the majority of the frameworks are based on LTL/CTL, mostly they deal with ‘*structural compliance*’ only. Also, they are able to provide the reasoning support for basic deontic notions; violations and CTD obligations are not supported. For example, Declare and BPMN-Q can only model achievement and prohibitions with the exception BPMN-Q is able to handle violations. More recently, COMPAS framework has been extended with new CRL patterns that enables the CMF to represent maintenance and contrary-to-duty obligations (Elgammal et al. 2014); however, COMPAS still has limited scope in representing permissions and perdurant obligations. In contrast, SeaFlows, which is based on first-order-logic (FOL), can only model achievement and prohibitions.

As for automated compliance checking, it is generally highly desirable that a formal language for compliance covers most of the properties, and properties of the environment of the unit under verification—for example normative requirements (Awad 2010). In addition, it should also support the complex properties from simpler ones. Most of the existing CMFs are based on temporal logic, as the maturity of temporal logic is undisputable. This is because the logic has strong foundations of automated and sophisticated analysis and verification tools, which have been successfully applied over two decades to verify large-scale industrial applications including safety and mission critical systems. However, temporal logic has limited reasoning capabilities for legal norms because, it has no conceptual relative correspondence to the legal domain; thus, the CMFs grounded on LTL cannot expressively model the properties of the norms. The authors in (Governatori 2015; Governatori and Hashmi 2015) provide fitting examples where not paying attention to legal reasoning principles leads to results contrary to those that legally trained professionals would produce—especially, the use of temporal logic produces results that lead to paradoxes giving conflicting and inaccurate reasoning results for modeling a specific type of norm. Hence, resolving the paradoxes is paramount without which it would not be possible to get desired results from compliance checking approaches that uses them or design new modeling languages that do not suffer from these paradoxes.

EC and FOL, on the other hand, also have their own limitations when it comes to providing reasoning and modeling support for all types of obligations modalities. For example, Hashmi et al. (2014) discusses the reasons why EC fails to capture the effects of temporal properties of norms; thus, concludes that EC is not suitable to reason about the deontic properties including capturing the effects of violations of obligation modalities. Whereas Herrestad (1991) concluded that FOL is not suitable to model norms. Norms formalised with FOL may have inconsistencies between the propositions of the rules. In some situations, one might get redundant information especially where rules are violated, and depending on the conditions new rules get the effects that might convey new information which is not available

in other propositions of the rules. Hence, the reasoning results produced by FOL cannot be relied upon for legal reasoning.

The evaluation results portray a somewhat *bleak picture* when it comes to seeing how existing frameworks represent legal knowledge for compliance checking because none is able to support all types of normative requirements. Primarily this is because of the formal language each framework uses to model the norms. However, this would not necessarily mean that the framework does not have expressive power to model the notion, but that the concept is not considered or analysed in that framework—including, the cases where the deontic concepts cannot be faithfully represented. This implies that adopting formalisms that are not conceptually grounded in legal practice creates a framework that is unreliable, and not suitable to be used in real-life applications.

As the regulations are generally frequently changed i.e., new rules are introduced, old are removed or updated—the changes in the regulations might prescribe new and more complex types of norms introducing new complexities. Then the question would be *how to model new types of norms when there need be?* This might introduce the researchers with a continuous dilemma of designing new modeling languages that are expressive, scalable and flexible to faithfully model ever changing normative requirements.

The real effectiveness of a CMF can depend on many factors, features and interfaces offered by the CMF. Since logic-based norms modeling languages are complex and often frustrate non-technical users; for example, business analysts or legal experts who lack the knowledge of formal and mathematical theories; the usability, comprehensibility and understandability of such formal languages is a major concern. Because it is not possible to use sophisticated verification tools—for example, model checking tools cannot be used without the use of a formal language. To surmount the usability issue there are number of proposals offering more graphical/patterns based languages that are more easily understood by the non-technical experts for modeling the compliance requirements as used in the evaluated CMFs. But the fundamental question is *whether the patterns used in these CMFs enhance the usability and understandability of the non-technical experts?* Indeed, the visual notation referring a modeling pattern can be more easily understood; however, what about the pattern itself? Does the pattern provide a good fit between the visual notation and the semantic concepts it refers to—in order to give a clear semiotic representation? Is there a balance between a pattern, the graphical symbol and their corresponding concepts? Meaning that the graphical notation used to represent a specific pattern really captures the intuition (or meanings) of the concept it represents. Currently, to the best of our knowledge, no study or empirical evaluation has been carried out in a scientific and rigorous manner on the usability, understandably and effectiveness of CMFs. At best, some anecdotic evidence is reported.

5 Related work

The evaluations presented in this chapter are comparable to several existing evaluations reported in the literature. The evaluations reported in Becker et al. (2012) cover the generalisability and applicability aspects of existing compliance management frameworks, and accumulates a detailed understanding on their implementation results. In evaluating the compliance rules and generalisability of the frameworks, they used a narrow and medium rules generalisability criteria, thus restricting their survey only to the checking of simple and complex compliance patterns representing the compliance requirements.

El Kharbili (2012), on the other hand, analyse operational and (non-)functional aspects of regulatory compliance management (RCM) from a business process management perspective using a three categories evaluation criteria. In the first category, the authors evaluate the RCM solutions from the business users, and the methodology and the architecture of the RCM perspective. In the second category, they evaluate nine functional areas of the RCM from a BPM perspective (for example, the strategy model, the business process model), and compliance dimensions such as compliance enforce, audit, and verification. In the last category, they use the functional and non-functional capabilities as the evaluation criteria. From the compliance dimensions, they extract three distinct types of rules—*structural*, *temporal* and *contractual* rules that are supported by the modeling languages. However, their comparative evaluation does not systematically evaluate “*legal requirements*” from the reasoning and proper modeling of the legal requirements perspective for compliance checking purposes. Also, they do not consider specific types of legal requirements and how they can be modelled. Whereas the work by Cabannilas et al. (2010) examines various CMFs with the focus on *how* they represent different types of business processes and regulatory rules using which formal languages to model them.

Otto and Anton (2007) examine various approaches to regulation modeling languages, and the extraction of key legal concepts from legal documents. Whereas Elgammal et al. (2011) reports a comparative analysis of three modeling languages for representing the compliance requirements with the focus on presenting these requirements at design-time. Their analysis covers the advantages and shortcomings to the selected languages selected from temporal and deontic families of logic, and the list 11 different features that a modeling language should have to provide formal specifications of various types of norms. In contrast, Turki and Bjekovic-Obradovic (2010) studies various approaches for achieving and maintaining the compliance of regulatory rules. Their work is rather similar to the one conducted by Otto and Anton (2007). The studies reported in Ly et al. (2013, 2015) evaluate five CMFs from health, manufacturing, maritime safety and IT project management domains using a predefined set of evaluation criteria derived from literature. However, their work lacks a detailed and comparative analysis of compliance modeling languages and modeling constructs for the specifications of norms. In contrast, Bonatti et al. (2004) investigates formal languages especially logic-based policy, trust

management, action, business rules and controlled natural languages for their possible usage for the specification of compliance requirements in security and policy domain.

The evaluation reported in this paper is complementary and different from these studies as it contributes a detailed understanding on various issues with existing CMFs—especially, from the modeling of the normative requirements perspective. However, it is different from them in the sense that we evaluate existing CMFs in order to examine *what they can do* to provide a comprehensive compliance management support, and *what modeling constructs* they provide to represent different classes of norms which has been largely ignored in these studies. The work of Elgammal et al. (2011) is somewhat close to the above-presented evaluation; however, their work is more generic in the sense that the authors examine *how the specifications of compliance requirements can be represented by a modeling language*, whereas we examine *what* specific constructs are provided by the CMF for modeling a specific type of norm. Furthermore, their comparative analysis is limited to the temporal and deontic families of logics. In contrast, we also evaluate the modeling constructs of the CMFs based on EC and FOL on top of modeling constructs of the LTL and Deontic logic-based CMFs. Moreover, by using the classes of norms of Sect. 2 we also examined whether the evaluated CMFs are able to provide a comprehensive modeling support for various classes of norms or not. It is worth noting that the analysis reported in this paper, in particular when a CMF does not support some particular type of norm, does not imply that the underlying formalism does not have the expressive power to model the corresponding notion. It means that the CMF does not expose such capabilities to the end-users (e.g., business analysts, lawyers, and other compliance professionals). Furthermore, in some cases, it is an open question whether some formalisms are able to model norms. For example, while Governatori (2015) identified some shortcomings of temporal logic representation of norm and Governatori and Hashmi (2015) exposed the problem for temporal logic based CMFs, the result does not state an impossibility theorem. Indeed, Governatori (2015) showed a method to correctly encode compliance requirements in temporal logic. The method assumes an oracle to determine whether a process is compliant and build back the temporal logic representation of the requirements. But such method is not useful for checking the compliance of business processes, since it requires the external oracle to first determine the compliance of the process against the requirements before building the corresponding formalisation. Currently, it is unknown whether temporal logics are able to provide appropriate formalisations without the need of an external oracle.

6 Conclusions

We investigated whether existing CMFs are able to provide modeling and reasoning support for various types of normative requirements and presented a methodological evaluation of seven selected CMFs using the conceptual model presented in Sect. 2. We primarily examined the conceptual foundations of the selected CMFs under predefined evaluation criteria and the obligation modalities representing the classes

of our classification model of normative requirements proposed elsewhere. Specifically we looked at the conceptual approaches that existing CMFs used to deal with the normative requirements related to the regulatory compliance. The reported evaluation is complementary and different from existing evaluations as we evaluated one-to-one mappings between different types of norms and the modeling constructs provided by a formal language used in a CMF. In addition, we also looked at what existing CMFs can do in terms of providing a comprehensive compliance management support; and whether they are able to provide a comprehensive reasoning and modeling support for various classes of norms or not.

The results of this evaluation reveal that none of the existing CMFs seem to be able to fully model different types of norms with the exception of PCL which provides the modeling operators that can be used to represent all classes of the norms of our classification model. This is because the use of the combination of deontic and defeasible logic increases the expressiveness of PCL while other CMFs do not provide any suitable constructs to represent some notions such as preemptiveness or perdurance effects of the norms. This raises serious questions on the effectiveness of existing CMFs and highlights exigent need for new modeling languages with sound theoretical and formal foundations to represent norms or address the problems with existing ones. Thus, the compliance results produced by a CMF can be relied upon if we want to model them with these languages.

Possible extensions The work presented here can be extended in a number of directions. The first possibility, as we previously mentioned (cf. Sect. 1.1), is that the formal semantics analysis of the norms modeling languages merits to be investigated separately, which can provide a better understanding on the strengths and weaknesses of such modeling languages. We are currently working in this direction. In Hashmi et al. (2014) we conducted a formal semantics analysis of PENELOPE and addressed the problems with its underlying formalism EC by introducing new predicates and events to capture various notions of norms. In addition, in Governatori and Hashmi (2015) we formally proved that modeling the norms with COMPAS's CRL patterns produce wrong compliance results—and we aim to continue this work.

Since, we have evaluated only the selected CMFs and the modeling constructs proposed in those CMFs. Besides these CMFs there are other modeling languages that are independent of any formalism and provide modeling constructs to represent norms. For example, N6mos 3 (Ingolfo et al. 2014) a *primitive*-based compliance verification language that uses primitives as notations to design graphical models to reason about the compliance requirements and the roles with the norms and *conceptual graph-based* norms modeling language (Croitoru et al. 2012). Further evaluations of such formalism independent languages can certainly provide further insights into the state-of-the-affairs as well as shortcomings of such languages.

Another line of interest can be gaining a better understanding of the concerns on the usability of the norms modeling constructs evaluated in this work—in particular, semiotic clarity of the concepts and the proposed patterns. A first step in this direction can be a detailed usability study with the non-technical experts and industry professionals that can provide more insights on these issues. A theoretical framework proposed by Figl et al. (2009) assessing the usability concerns from a

cognitive quality aspect(s) of process modeling languages such as perceptual immediacy, graphical parsimony etc., can be benefited as a guiding framework to gain more understanding on the balance between the complexity and expressiveness of the modeling constructs and logic formulas for conducting such usability studies.

Acknowledgements This is a revised and extended version of AP-BPM 2013 paper (Hashmi and Governatori, 2013). We thank Ho-Pun Lam and Régis Riveret for their valuable comments on the draft of this paper.

References

- Abdullah NS, Sadiq S, Indulska M (2010) Emerging challenges in information systems research for regulatory compliance management. In: Proceedings of the 22nd international conference on advanced information systems engineering. CAiSE'10. Springer, pp 251–265
- Allaire M, Governatori G (2014) On the equivalence of defeasible deontic logic and temporal defeasible logic. In: Dam H, Pitt J, Xu Y, Governatori G, Ito T (eds) PRIMA 2014: principles and practice of multi-agent systems, vol 8861. LNCS. Springer, pp 74–90. doi:[10.1007/978-3-319-13191-7_7](https://doi.org/10.1007/978-3-319-13191-7_7)
- Antoniou G, Billington D, Governatori G, Maher MJ (2001) Representation results for defeasible logic. ACM Trans Comput Log 2(2):255–287. doi:[10.1145/371316.371517](https://doi.org/10.1145/371316.371517)
- Arbab F (2004) REO: a channel-based coordination model for component composition. Math Struct Comput Sci 14(3):329–366
- Awad A (2007) BPMN-Q: a language to query business processes. In: Enterprise modelling and information systems architectures—concepts and applications: proceedings of the 2nd international workshop on enterprise modelling and information systems architectures (EMISA'07). St. Goar, Germany, 8–9 Oct 2007, pp 115–128
- Awad A (2010) A compliance management framework for business process models. Ph.D. thesis, Hasso Plattner Institute, Potsdam University, Germany
- Awad A, Weidlich M, Weske M (2011) Visually specifying compliance rules and explaining their violations for business processes. J Vis Lang Comput 22(1):30–55
- Awad A, Decker G, Weske M (2008) Efficient compliance checking using BPMN-Q and temporal logic. In: BPM. LNCS. Springer, pp 326–341
- Awad A, Polyvyanyy A, Weske M (2008) Semantic querying of business process models. In: 12th international IEEE on enterprise distributed object computing conference, 2008. EDOC '08, pp 85–94. doi:[10.1109/EDOC.2008.11](https://doi.org/10.1109/EDOC.2008.11)
- 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
- Bandara W, Miskon S, Fietl E (2011) A systematic, tool-supported method for conducting literature reviews in information systems. In: Virpi T, Joe N, Matti R, Wael S (eds) Proceedings of 19th European conference on information systems. ECIS 2011, Helsinki
- Baral C, Zhao J (2007) Non-monotonic temporal logics for goal specification. In: Proceedings of the 20th international joint conference on artificial intelligence (IJCAI 2007). Morgan Kaufmann Publishers Inc, pp 236–242
- BCBS (2013) Basel III: the liquidity coverage ratio and liquidity risk monitoring tools. <http://www.bis.org/publ/bcbs238.pdf>
- Becker M, Laue R (2012) A comparative survey of business process similarity measures. Comput Ind 63(2):148–167
- 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 Bus Res J 5(2):221–247
- Bonatti PA, Shahmehri N, Duma C, Olmedilla D, Nejdil W, Baldoni M, Baroglio C, Martelli A, Coraggio P, Antoniou G, Peer J, Fuchs NE (2004) Rule-based policy specification: state of the art and future work. REWERSE Project Report-i2-D1. Report, Università di Napoli Fedrecio II
- 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

- Croitoru M, Oren N, Miles S, Luck M (2012) Graphical norms via conceptual graphs. *Knowl Based Syst* 29:31–43
- 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 MS (2009) Business compliance governance in service-oriented architectures. In: International conference on advanced information networking and applications, 2009. AINA '09, pp 113–120
- Dwyer M, Avrunin G, Corbett J (1999) Patterns in property specifications for finite-state verification. In: Proceedings of the 1999 international conference on software engineering, 1999, pp 411–420
- El Kharbili M (2012) business process regulatory compliance management solution frameworks: a comparative evaluation. In: APCCM 2012, CRPIT 130, pp 23–32
- Elgammal AFSA (2012) Towards a comprehensive framework for business process compliance. Ph.D. thesis, Tiburg University. <https://ideas.repec.org/p/tiu/tiutis/a30c4513-4b19-44f1-beb0-00b3c2d6f15e.html>
- Elgammal A, Turetken O, van den Heuvel WJ, Papazoglou M (2011) On the formal specification of regulatory compliance: a comparative analysis. In: Proceedings of ICSOC'10, pp 27–38
- Elgammal A, Turetken O, Heuvel WJ, Papazoglou M (2014) Formalizing and applying compliance patterns for business process compliance. *Softw Syst Model* 15(1):119–146. doi:10.1007/s10270-014-0395-3
- Elgammal A, Turetken O, van den Heuvel WJ, Papazoglou MP (2010) Root-cause analysis of design-time compliance violations on the basis of property patterns. In: ICSOC, pp 17–31
- FATF (2017) The FATF recommendations: international standards on combating money laundering and the financing of terrorism and proliferation. <http://www.fatf-gafi.org/publications/fatfrecommendations/documents/fatf-recommendations.html>
- Figl K, Mandling J, Strembeck M (2009) Towards a usability assessment of process modelling languages. In: Markus N, Rump F, Jan M, Nick G (eds) Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2009), Ceur workshop proceedings, vol 554, pp 138–156 <http://ceur-ws.org/Vol-554/epk2009-paper09.pdf>
- Fongon P, Grillo K (2004) Corporate implications of Sarbanes Oxley Act: a public policy. <http://www.global-trade.law.com/ITRN711>
- Ghose A, Koliadis G (2007) Auditing business process compliance. In: Krämer B, Lin KJ, Narasimhan P (eds) Service-oriented computing (ICSOC 2007), vol 4749. LNCS. Springer, pp 169–180
- Giblin C, Liu AY, Müller S, Pfizmann B, Zhou X (2005) Regulations expressed as logical models (REALM). In: Proceeding of the 18th annual conference on legal knowledge and information systems (JURIX 2005). IOS Press, pp 37–48
- Goedertier S, Vanthienen J (2006) Compliant and flexible business processes with business rules. In: BPMDS, vol 236. CEUR workshop proceedings, CEUR-WS.org
- Goedertier S, Vanthienen J (2006) Designing compliant business processes with obligations and permissions. In: Eder J, Dustdar S (eds) Business process management workshops 2006. LNCS 4103. Springer, pp 5–14
- Governatori G (2015) Thou shalt is not you will. In: Atkinson K (ed) Proceedings of the fifteenth international conference on artificial intelligence and law. ACM, New York
- Governatori G (2005) Representing business contracts in RuleML. *Int J Cooper Inf Syst* 14(2–3):181–216
- Governatori G, Rotolo A (2006) Logic of violations: a Gentzen system for reasoning with contrary-to-duty obligation. *Australas J Log* 4:193–215
- Governatori G, Hashmi M (2015) No time for compliance. In: Proceedings of 19th IEEE the enterprise computing conference (EDOC'15)
- Governatori G, Milosevic Z, Sadiq S (2006) Compliance checking between business processes and business contracts. In: 10th international enterprise distributed object computing conference (EDOC 2006). IEEE Computing Society, pp 221–232
- Governatori G, Rotolo A (2010) A conceptually rich model of business process compliance. In: Proceedings of APCCM '10, vol 110, pp 3–12
- Governatori G, Sadiq S (2009) The journey to business process compliance. In: Cardoso J, van der Aalst W (ed) Handbook of research on business process management, Chap 20. IGI Global, pp 426–454. doi:10.4018/978-1-60566-288-6.ch020
- Hashmi M, Governatori G, Wynn MT (2016) Normative requirements for regulatory compliance: an abstract formal framework. *Inf Syst Frontiers* 18(3):429–455. doi:10.1007/s10796-015-9558-1
- Hashmi M, Governatori G, Wynn MT (2012) Business process data compliance. In: Proceedings of 6th international symposium. RuleML 2012, Montpellier, pp 32–46

- Hashmi M, Governatori G, Wynn MT (2013) Normative requirements for business process compliance. In: Proceedings of 3rd symposium (ASSRI'13) on service research and innovation, Sydney, pp 100–116
- Hashmi M, Governatori G, Wynn MT (2014) Modeling obligations with event-calculus. In: Proceedings of 8th international symposium. RuleML 2014, Prague., pp 296–310
- Herrestad H (1991) Norms and formalization. In: Proceedings of ICAIL 1991, pp 175–184
- 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
- HIPAA TUG (1996) The US Health Insurance Portability and Accountability Act of 1996
- IFRS (2014) IFRS 7 international financial reporting standards: financial instruments disclosures. <http://www.ifrs.org/IFRSs/Pages/IFRS.aspx>
- Ingolfo S, Jureta I, Siena A, Perini A, Susi A (2014) Nmos 3: legal compliance of roles and requirements. In: Yu E, Dobbie G, Jarke M, Purao S (eds) Conceptual modeling, vol 8824. Lecture Notes in Computer Science. Springer, pp 275–288
- Johansson LO, Wårja M, Carlsson S (2012) An evaluation of business process model techniques, using Moody's quality criterion for a good diagram. In: BIR12, vol 963. CEUR workshop proceedings, CEUR-WS.org
- Karagiannis D (2008) A business process-based modeling extension for regulatory compliance. In: Multikonferenz Wirtschaftsinformatik
- Każmierczak P, Pedersen T, Ågotnes T (2012) NORMC: a norm compliance temporal logic model checker. In: STAIRS 2012 - Proceedings of the sixth starting AI researchers' symposium, Montpellier, France, 27–28 August 2012, vol 241. IOS Press, pp 168–179. doi:[10.3233/978-1-61499-096-3-168](https://doi.org/10.3233/978-1-61499-096-3-168)
- Lu R, Sadiq S (2007) A survey of comparative business process modeling approaches. In: Abramowicz W (ed) Business information systems, vol 4439. LNCS. Springer, Heidelberg, pp 82–94
- Lu R, Sadiq S, Governatori G (2007) Compliance aware business process design. In: 3rd international workshop on business process design (BPD'07). Springer, pp 120–131
- Ly LT, Knuplesch D, Rinderle-Ma S, Goesser K, Reichert M, Dadam P (2010) SeaFlows toolset—compliance verification made easy. In: CAiSE'10 Demos
- Ly LT, Maggi FM, Montali M, Rinderle S, van der Aalst W (2013) A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: Proceeding of EDOC
- Ly L, Rinderle-Ma S, Dadam P (2010) Design and verification of instantiable compliance rule graphs in process-aware information systems, vol 6051. Springer, Berlin, pp 9–23. doi:[10.1007/978-3-642-13094-6_3](https://doi.org/10.1007/978-3-642-13094-6_3)
- Ly LT, Rinderle-Ma S, Göser K, Dadam P (2012) On enabling integrated process compliance with semantic constraints in process management systems. Inf Syst Frontiers 14(2):195–219
- Ly LT, Maggi FM, Montali M, Rinderle S, van der Aalst W (2015) Compliance monitoring in business processes: functionalities, application, and tool-support. Inf Syst. doi:[10.1016/j.is.2015.02.007](https://doi.org/10.1016/j.is.2015.02.007)
- Ly L, Rinderle-Ma S, Knuplesch D, Dadam P (2011) Monitoring business process compliance using compliance rule graphs. In: Meersman R, Dillon T, Herrero P, Kumar A, Reichert M, Qing L, Ooi BC, Damiani E, Schmidt D, White J, Hauswirth M, Hitzler P, Mohania M (eds) On the move to meaningful internet systems: OTM 2011, vol 7044. LNCS. Springer, Berlin, pp 82–99
- Maggi F, Montali M, Westergaard M, van der Aalst W (2011) Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: BPM. LNCS 6896. Springer, pp 132–147
- Maggi F, Westergaard M, Montali M, van der Aalst W (2011) Runtime verification of LTL-based declarative process models. In: Proceedings of RV. LNCS. Springer
- Makinson D, van der Torre L (2003) Permission from an input/output perspective. J Philos Log 32(4):391–416
- MASTER (2008) Managing assurance, security, and trust for services. FP7-ICT integrated project for secure, dependable, and trusted infrastructures
- McIntyre SR (2008) Integrated governance, risk and compliance: improve performance and enhance productivity in federal agencies. Technical report, PricewaterhouseCoopers
- Mili H, Tremblay G, Jaoude GB, Lefebvre E, Elabed L, Boussaidi GE (2010) Business process modeling languages: sorting through the alphabet soup. ACM Comput Surv 43(1):1–56. doi:[10.1145/1824795.1824799](https://doi.org/10.1145/1824795.1824799)
- Montali M (2010) Specification and verification of declarative open interaction models: a logic-based approach, vol 56. LNBIP. Springer, Berlin

- Montali M, Pesic M, van der Aalst WMP, Chesani F, Mello P, Storari S (2010) Declarative specification and verification of service choreographiess. *ACM Trans Web* 4(1):3:1–3:62
- Olivieri F (2014) Compliance by design. Synthesis of business processes by declarative specifications. Ph.D., Dipartimento di Informatica, Università degli Studi di Verona, Italy and Institute for Integrated and Intelligent Systems, Griffith University, Australia
- Otto P, Anton A (2007) Addressing Legal requirements in requirements engineering. In: 15th IEEE international on requirements engineering conference, 2007. RE '07, pp 5–14
- Palmirani M, Governatori G, Contissa G (2011) Modelling temporal legal rules. In: Proceedings of the 13th international conference on artificial intelligence and law (ICAIL 2011). ACM Press
- Pesic M, Schonenberg H, van der Aalst W (2007) DECLARE: full support for loosely-structured processes. In: Proceedings of 11th IEEE international conference on enterprise distributed object computing (EDOC'07), pp 287–287
- Pesic M, van der Aalst W (2006) A declarative approach for flexible business processes management. In: BPM workshops, vol 4103. LNCS. Springer, pp 169–180
- Ramezani E, Fahland D, van der Werf J, Mattheis P (2012) Separating compliance management and business process management. In: Daniel F, Barkaoui K, Dustdar S (eds) Business process management workshops, vol 100. LNBIP. Springer, Berlin, pp 459–464. doi:[10.1007/978-3-642-28115-0_43](https://doi.org/10.1007/978-3-642-28115-0_43)
- Ramezani E, Fahland D, van der Aalst W (2012) Where did i misbehave? Diagnostic information in compliance checking. In: Proceedings of business process management, pp 262–278
- Ramezani E, Fahland D, van Dongen BF, van der Aalst W (2013) Diagnostic information for compliance checking of temporal compliance requirements. In: CAiSE, pp 304–320
- Rieke R, Repp J, Zhdanova M, Eichler J (2014) Monitoring security compliance of critical processes. In: 2014 22nd Euromicro international conference on parallel, distributed and network-based processing (PDP), pp 552–560
- Sadiq S, Governatori G (2015) Managing regulatory compliance in business processes. In: vom Brocke J, Rosemann M (eds) Handbook of business process management, vol 2, 2nd edn. International handbooks on information systems. Springer, Berlin, pp 265–288
- Sadiq S, Governatori G, Namiri K (2007) Modeling control objectives for business process compliance. In: Proceedings of BPM'07. Springer, pp 149–164
- Sartor G (2005) Legal reasoning: a cognitive approach to the law. Springer, Berlin
- SCBS (2004) BASEL II accord - International convergence of capital measurement and capital standards: a revised framework. <https://www.federalreserve.gov/boarddocs/press/bcreg/2004/20040626/attachment.pdf>
- Schumm D, Turetken O, Kokash N, Elgammal A, Leymann F, Heuvel WJVD (2010) Business process compliance through reusable units of compliant processes. In: Proceedings of international conference on current trends in web engineering
- Türetken O, Elgammal A, van den Heuvel WJ, Papazoglou M (2012) Capturing compliance requirements: a pattern-based approach. *Softw IEEE* 29(3):28–36. doi:[10.1109/MS.2012.45](https://doi.org/10.1109/MS.2012.45)
- Türetken O, Elgammal A, van den Heuvel WJ, Papazoglou M (2011) Enforcing compliance on business processes through the use of patterns. In: Proceeding of European conference on information system. <http://aisel.aisnet.org/ecis2011/5>
- Turki S, Bjekovic-Obradovic M (2010) Compliance in e-government service engineering: state-of-the-art. In: Exploring services science. LNBIP. Springer, pp 270–275
- US-Government (2002) Public Company Accounting Reforms and Investor Protection Act (Sarbanes-Oxley Act). Public Law 107-204, 116 Stat. 745
- van der Aalst W, Pesic M, Schonenberg H (2009) Declarative workflows: balancing between flexibility and support. *Comput Sci Res Dev* 23:99–113
- van der Aalst W, ter Hofstede A, Kiepuszewski B, Barros A (2002) Workflow patterns. QUT Technical report. FIT-TR-2002-02, Queensland University of Technology, Brisbane, Australia. <http://www.workflowpatterns.com/documentation/documents/wfs-pat-2002.pdf>