Supporting Domain Experts to Select and Configure Precise Compliance Rules

Elham Ramezani(✉), Dirk Fahland, and Wil M.P. van der Aalst

Eindhoven University of Technology, Eindhoven, The Netherlands
{e.ramezani,d.fahland,w.m.p.v.d.aalst}@tue.nl

Abstract. Compliance specifications concisely describe selected aspects of what a business operation should adhere to. To enable automated techniques for compliance checking, it is important that these requirements are specified correctly and precisely, describing exactly the behavior intended. Although there are rigorous mathematical formalisms for representing compliance rules, these are often perceived to be difficult to use for business users. Regardless of notation, however, there are often subtle but important details in compliance requirements that need to be considered. The main challenge in compliance checking is to bridge the gap between informal description and a precise specification of all requirements. In this paper, we present an approach which aims to facilitate creating and understanding formal compliance requirements by providing configurable templates that capture these details as options for commonly-required compliance requirements. These options are configured interactively with end-users, using question trees and natural language. The approach is implemented in the Process Mining Toolkit ProM.

Keywords: Compliance specification · Compliance checking · Configurable compliance rules · Auditing · Question tree

1 Introduction

Compliance checking techniques determine if business operations are within the boundaries set by law, managers and other stakeholders or obey security requirements set by the company. Such constraints can be formalized using different specification formalisms such as temporal logic [14] or deontic logic [30] depending on the compliance checking technique that is being employed. A problem often encountered in practise [19], however, is specifying precisely the behavior intended.

Many practitioners prefer capturing compliance requirements using informal notations, such as natural language, instead of formal specification languages. These representations are more accessible but often imprecise and of less value when doing automated compliance checking. Since domain experts usually describe informally a compliance requirement, technical experts may invest considerable effort formalizing it and check if the recorded process executions conform with it, only to later determine that the property has been

N. Lohmann et al. (Eds.): BPM 2013 Workshops, LNBIP 171, pp. 498–512, 2014.
DOI: 10.1007/978-3-319-06257-0_39,

specified incorrectly. Whereas if domain experts are involved in the specification process, the intended behavior with all its subtle aspects can be specified directly and thus avoiding ambiguities.

Numerous researchers have developed specification patterns to facilitate construction of formal specification of compliance requirements. Feedback indicates [16] that these patterns are considered helpful but they fail to capture subtle aspects of a specific requirement. In addition, adaption and application of these patterns are not trivial for many practitioners as they are less familiar with the underlying formalization.

This paper describes an approach that addresses the gap between informal requirements and formal compliance specifications. We introduce an interactive approach for using tacit knowledge of domain experts to specify compliance requirements. Our approach aims at (i) enabling business users and compliance experts to specify compliance constraints and (ii) encouraging them to think about the subtle aspects of their intended behavior when specifying a constraint. The key components of this process are *question trees*, and *configurable generic compliance patterns* pre-formalized in configurable Petri nets that capture common compliance requirements. We have developed a repository of configurable compliance patterns. Every pattern allows for alternative variations of a compliant behavior. Selecting an appropriate configurable pattern and configuring a pattern for its configuration options are done interactively with user. A questionnaire consisting of two question trees asks users about their intended compliant behavior. The first question tree helps the user selecting a general compliance requirement, i.e., a configurable pattern. The second tree helps the user configuring a general requirement w.r.t. various subtle semantic aspects. The approach is implemented and a case study is being prepared to evaluate the approach.

The remainder of this paper is organized as follows. Section 2 explains a compliance management life cycle. An overview of the methodology and notions that our work is built on are discussed in Sect. 3. Section 4 introduces the repository of configurable compliance patterns. Section 5 describes how this approach facilitates compliance specification for domain experts and showcases implementation of the technique in ProM. We will review the related work in Sect. 6 and finally Sect. 7 concludes the paper and motivates future work.

2 Compliance Management

Organizations are confronted with an ever growing set of laws and regulations to comply to. Failing to comply to regulations can impose severe risks such as penal consequences on management level or lost contracts with clients. *Compliance Management* (CM) within an organization comprises the design, implementation, maintenance, verification and reporting of compliance requirements and it calls for a structured methodology. We proposed a compliance management life cycle in [23] as a methodology to elicit, formalize, implement, check, and optimize compliance requirements in organizations. As is shown in Fig. 1, compliance management activities can be identified as:

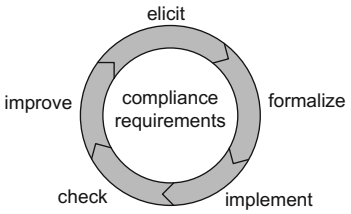

Fig. 1. Compliance management life cycle

- *Compliance Elicitation*: determine the compliance requirements that need to be satisfied. (i.e., rules defining the boundaries of compliant behavior).
- *Compliance Formalization*: specify formally compliance requirements originating from laws and regulations derived in the compliance elicitation phase.
- *Compliance Implementation*: enforce specified compliance requirements in business operation.
- *Compliance Checking*: investigate whether the constraints will be met (forward compliance checking) or have been met (backward compliance checking).
- *Compliance Optimization*: improve business processes and their underlying information systems based on the diagnostic information gained from compliance checking.

In the following we will elaborate on elicitation and formalization and briefly discuss compliance checking.

Compliance Elicitation and Formalization. Specifying precise compliance requirements spans over *Compliance Elicitation* and *Compliance Formalization* phases of the CM life cycle and introduces many challenges. It calls for combination of different knowledge areas such as compliance expertise, formalization skills, and domain specific knowledge.

Regulations are usually presented informally and described in an abstract way because they need to be independent from implementation. Moreover, the writers and users of regulations are lawyers or business users, their instrument of work uses natural language. This language is non-formalized and incorporates domain specific terminology, as well as structure and definitions. Therefore enforcing and checking a compliance requirement requires a precise formalization of this requirement. *In the step from natural language to precise formalization many subtle aspects of the requirement have to be considered.*

For instance, consider a compliance requirement we obtained from internal policies of a specialized hospital that accepts only patients requiring a specific medical treatment: "For every patient registered in the hospital an X-ray must be taken". This compliance requirement enforces that *patient registration* must be followed by activity *X-ray*. The requirement seems very straightforward but no matter which formalism is chosen for this simple requirement, while formalizing, it is important to decide about some details e.g.,: (1) whether *patient registration* should be directly followed by *X-ray* or other activities may occur in between the specified sequence; (2) whether it is allowed that other activities occur before

patient registration or a patient cannot receive any treatment without registration; (3) whether a patient can be registered several times (for instance in different departments) and if yes; (4) should the specified sequence be followed every time; (5) whether it is allowed that the specified sequence never occurs i.e., if it is allowed that a patient is never registered. Interpreting an informal rule with all its details can be surprisingly difficult and must be done by domain experts who are usually less familiar with different formalisms. Therefore an approach is required to hide the complexity of formalization from business user and at the same time support automated compliance checking. In this context an interactive *'question and answer'* approach based on "disciplined" natural language seems promising. Such an approach is used in property specification for software development in [8,19,28] and is a suitable candidate for compliance specification. However, compliance specification is more challenging as, unlike in software development, the formalized requirement is not inspected again by an expert in formal techniques and immediately used to check compliance.

Compliance Checking. Precisely formulated compliance requirements derived from previous phases in CM life cycle are used for verification, monitoring and auditing of business processes. There are two basic types of compliance checking: (1) forward compliance checking aims to design and implement processes where compliant behavior is enforced [6,12,13,18,26] and (2) backward compliance checking aims at detecting and localizing non-compliant behavior [2,5,17,25] that happened in the past. Regardless of which analysis technique is used, automated compliance checking can only be applied if a compliance requirement has been specified precisely.

Compliance Rule Repository. In [2,23] we have shown that compliance requirements (originating from legislations) restrict one or several perspectives of a process including control flow, data flow, process time or organizational aspects. In [20,22] we have shown how a complex compliance requirement covering several perspectives of a process can be decomposed into smaller compliance rules which can be formalized as parameterized compliance patterns in terms of Petri nets. These Petri nets then can be used in backward compliance checking to provide diagnostic information about compliance violations.

This approach is supported by a repository of more than 50 compliance patterns covering a majority of the compliance rules found in literature [21]. In this paper we present an approach to consolidate this repository and *to select and configure the right rule to precisely express a given informal description.*

3 Methodology

As is motivated in Sect. 2, compliance requirement specification calls for an approach that allows for defining different variations of a compliance requirement, and is accessible in order to benefit from the compliance expertise of business users and mathematically precise to enable automated compliance checking. That is, it needs to offer variations of a specified behavior, hide complexity

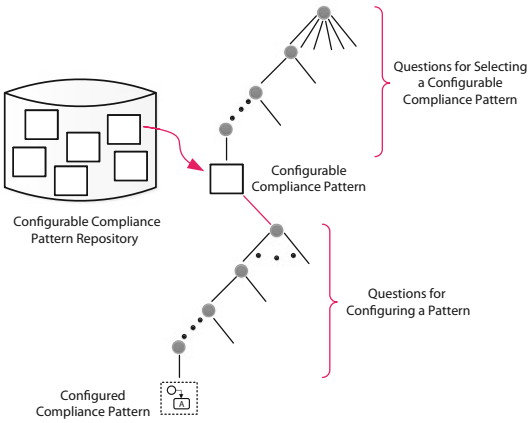

Fig. 2. Compliance specification overview

of formalization from business users and at the same time produce a formal definition of the compliance requirement.

In this section, we explain how our approach can help practitioners elucidate a compliance requirement by making informed choices between different variations of a compliance rule. Figure 2 gives an overview of our approach for compliance specification. This approach is built upon a repository of configurable compliance patterns.

Configurable Compliance Pattern Repository. Although the collection of compliance rules in [21] is comprehensive, there are subtle variations of a compliance requirement which cannot be expressed only by selecting a compliance rule from the rule repository and instantiating it for its parameters, rather slight modification in the underlying formalization may be necessary. Therefore one would like to see a general rule which allows to define all possible variations.

In addition there are over 50 compliance rules (only for control-flow perspective) in the rule repository which makes the choice of appropriate compliance rule cumbersome and error prone if the user is not familiar with the underlying formalization. To help the user selecting the right rule, we consolidated the compliance rules by merging similar rules (that differ in variations of subtle semantic aspects) into one configurable compliance pattern that is easier to describe in general terms. Consolidating similar rules into a configurable pattern is done manually following a generic approach. We first define a core behavior for the configurable pattern and then extend the core behavior with all possible configuration options. These configuration options allow to define different variations of a compliance requirement. The idea is that a user first picks a general configurable pattern with all its configuration options and then configures it w.r.t. various subtle aspects. Details of the repository of configurable patterns are given in Sect. 4.

Question Tree. In order to enable domain experts to specify the intended behavior of a compliance requirement, we apply an interactive question and answer based approach. We aim to guide users to select an appropriate configurable compliance pattern and elaborate on how to configure its configuration options such that it represents intended behavior. Thus we apply a Question Tree (QT) representation which is basically a decision tree and its content is based on disciplined natural language.

We apply *two* distinct question trees; a set of questions which guide the user to *select a specific configurable compliance pattern* and a set of questions which are asked to resolve different configuration options of a chosen configurable pattern in order to specify details of intended admissible behavior.

Questions to Select a Configurable Compliance Pattern. The QT of the first phase breaks the problem of deciding which configurable pattern is most appropriate by asking users to consider only one differentiating attribute at a time. In this phase, QT has a hierarchical structure and this structure supports the isolation of concerns, only presenting a question to the user that is relevant in context of their previous answer. A new question that can be revealed after answering a given question is a child question of that previous answer; the previous question is the parent question of that child question. By selecting a different answer to a parent question, the user will explore a different set of child questions that are relevant to that answer and will arrive at a different configurable pattern. Figure 3 QT-phase1 (left) presents the question tree for selecting a configurable pattern in the example discussed earlier in Sect. 2.

Questions to Configure a Configurable Compliance Pattern. Questions in the second phase concern configuring subtle behavioral aspects of a specific

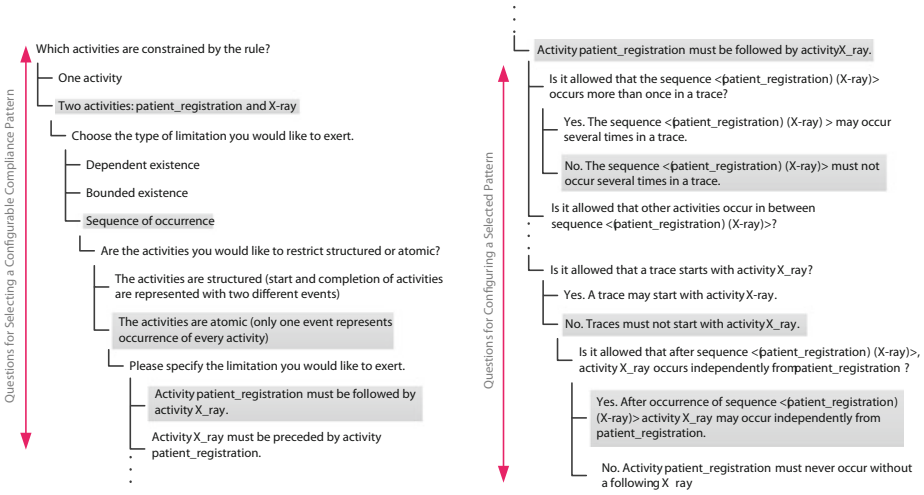

Fig. 3. QT-phase1 (left), QT-phase2 (right)

pattern. Not all questions in this phase have a hierarchical structure. That is, many questions in this phase can be asked in any order, since there are some options in each of configurable patterns which are conceptually orthogonal to each other. These questions will be presented to the user together and s/he may answer them in any order based on personal preferences and understanding. However, some options are not orthogonal e.g., a question whether a sequence of repeated events may occur several times is only meaningful if the user first answers that a sequence of repeated events is allowed. In such cases, the former question is only asked if a certain pre-configuration holds for it. Please note that the configurable pattern i.e., the underlying Petri net and its configuration options are *not shown* to the end user and user only deals with textual descriptions of rules in terms of questions and answers. In the back-end, every *answer node* of QT in the second phase is mapped to a configuration option in a configurable pattern and configures the pattern based on choices user makes. The configuration process is continued until all details of a compliant behavior is decided. Figure 3 QT-phase2 (right) presents partially the question tree of the second phase for the example of Sect. 2.

Illustrating a Compliance Rule to a Domain Expert. The configurable compliance pattern is hidden from user and s/he is only represented with questions and answers which are designed in a simple hence structured and clear text. In order to remove any ambiguity for the user while answering questions of subtle behavioral aspects, there are several compliant and non-compliant sample traces given for every answer. That is, a user can easily see how a certain choice can impact (i.e., limit or extend) admissible behavior. The configured compliance pattern determined in the second phase is a Petri net that can be used for automated compliance checking applying the techniques in [20,22].

In the following we will first discuss the repository of configurable compliance patterns and then show a walk-though example illustrating how a user selects and configures a compliance rule using the two question trees.

4 Consolidating and Organizing Compliance Rules in a Repository

The configurable compliance pattern repository is built upon the collection of control-flow compliance rules in [21]. We consolidated these rules by merging similar rules into a configurable pattern to eliminate redundancies and allow for specifying different variations of a rule. A configurable compliance pattern is a configurable Petri net which describes a group of compliance rules in a concise way. Originally configurable process models [3,24] were proposed to describe variants of a reference process. Here, we are applying the concept to describe variants of compliance requirements.

Every configurable compliance pattern is parameterized and formalized in terms of Petri nets with a core component. This core structure enforces a core behavior (e.g., a sequence). In addition a pattern has several other components

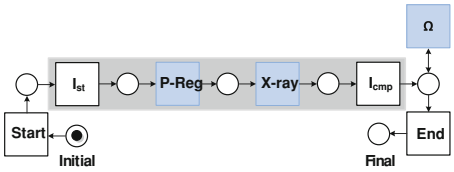

Fig. 4. Sequence of *P-Reg* and *X-ray*

which determine variations of core behavior. Core behavior enables a clear distinction between commonalities shared among compliance rules in one category and variability.

To consolidate the rules in [21], we studied rules which share a common behavior. We kept the core component in a configurable pattern and added all possible configuration options to it. The resulting configurable pattern can describe all the original rules it is derived from, and many more because of the new possible combination of different configuration options. The configurable patterns are sound be design. Please recall the example given earlier in Sect. 2. The Petri net pattern shown in Fig. 4 formalizes the core behavior of the requirement of this example.

The compliance pattern starts by firing transition *Start* and a token in place *Final* represents a completed case. The core of the rule is formalized in the grey-shaded part between transitions I_{st} and I_{cmp} which represents an instance of the compliance rule. The rule becomes active when I_{st} fires and it is satisfied when I_{cmp} fires. The hollow transitions (*Start*, I_{st}, I_{cmp}, and *End*) are invisible. The core structure of the pattern enforces; if *patient registration* (*P-Reg*) occurs then it must be followed by *X-ray*. Every compliance pattern allows to focus on activities restricted by the corresponding compliance rule and abstract from all other activities in a process. The Ω activity after I_{cmp} represents any other activity in a process apart from *P-Reg* and *X-ray*. If we want to add other options to the behavior specified in the Petri net pattern in Fig. 4, we need to add some more components to the pattern and build a configurable pattern out of it.

The configurable pattern shown in Fig. 5 is parameterized over the activity names such that activity A = *P-Reg* and activity B = *X-ray*. The configurable pattern allows for defining variations of the core behavior and by blocking or activating a component we can extend or limit admissible behavior. In the following we will explain the components of the configurable pattern in Fig. 5 and explain how blocking or activating a component can change the behavior of the pattern.

- *Comp.1-*Ω: Activating this component allows for occurrence of arbitrary other activities in between the sequence $\langle$*(P-Reg)(X-ray)*$\rangle$ and blocking this component enforces that activity *P-Reg* must be followed directly by *X-ray*.
- *Comp.2-*Ω: Activating or blocking this component, enforces that other activities may occur before *P-Reg* or not.

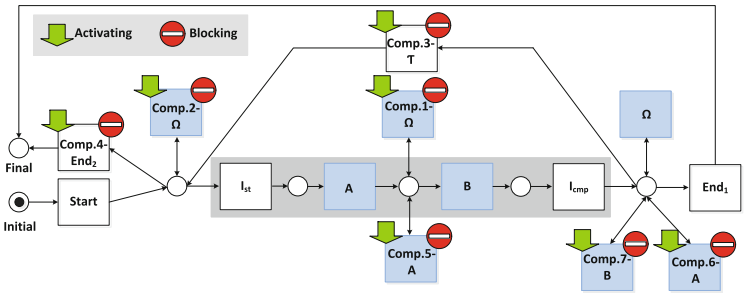

Fig. 5. Configurable sequence of *P-Reg* and *X-ray*

- *Comp.3-τ*: Activating or blocking this component allows that the sequence ⟨*(P-Reg)(X-ray)*⟩ occurs multiple times in a trace or not.
- *Comp.4-End*$_2$: Activating or blocking this component allows that a patient, would never get registered or not.
- *Comp.5-A*: Activating or blocking this component allows that several registrations of a patient can be followed by one execution of activity *X-ray* or not.
- *Comp.6-A*: Activating or blocking this component allows that after occurrence of the sequence ⟨*(P-Reg)(X-ray)*⟩ a patient gets registered without a following *X-ray* or not.
- *Comp.7-B*: Activating or blocking this component allows that activity *X-ray* occurs independently from the specified sequence of ⟨*(P-Reg)(X-ray)*⟩ or not.

When designing a configurable compliance pattern, we abstract from concrete examples and consider all possible configuration options. The configuration options we address in our approach include: *activating*, *blocking*, and *hiding/skipping* a transition, an arc or a group of transitions and arcs. In addition we consider configuring *arc weights*.

By developing configurable patterns, we could eliminate redundancies in a compliance rule family and reuse the commonalities, thus decreasing the number of patterns to 22 configurable compliance patterns having 0–38 configuration options each. This way, over 1000 different compliance patterns can be derived (including the original 50 patterns) though picking different configuration options.

5 Supporting Domain Experts to Specify Compliance Constraints

In this section we will elaborate our methodology and its implementation by going through a real life example step by step and showcase how a user who is not familiar with any formalism specifies his/her admissible behavior considering its detailed aspects.

The technique is implemented in the *Compliance* package of the Process Mining Toolkit ProM6, available from http://www.processmining.org. The package contains the repository of all configurable compliance patterns. The *Elicit Compliance Rule* plug-in takes a log as input and returns a compliance rule using the approach of Sect. 3. The returned rule can be used for compliance checking using the *Check Compliance of a Log* plug-in. In the following we show how a user can use this implementation to select and configure a compliance rule.

We chose the event log taken from *BPI Challenge 2011* available from [1]. The log is taken from a Dutch Academic Hospital. This log contains some 150.000 events in over 1100 cases. Apart from some anonymization, the log contains all data as it came from the hospital's systems. Each case corresponds a patient of the hospital's Gynaecology department. The log contains information about when certain activities took place, which group performed the activity and so on. Many attributes have been recorded that are relevant to the process.

To demonstrate the approach, we chose to formalize a rule that captures the following behavior observed on the event log [7]: *Glucose level must be estimated 4 times repetitively if a patient diagnosed for cervical cancer of uterus (diagnosis code M13) and classified as an urgent case*[1]. We have preprocessed this log for patients who are suffering from cervical cancer of uterus. Urgent patients are those cases where at least one activity of type urgent is manifested. A very common activity representing an urgent case is *'haemoglobin photoelectric-urgent'*. If we rephrase the constraint and substitute the activity names with corresponding event names in the log, the rule states: *In case of patients diagnosed for code M13, activity 'haemoglobin-photoelectric-urgent' must be followed 4 times by activity 'glucose-urgent'*.

We take this log as input and run the *Elicit Compliance Rule* plug-in that implements the approach of Sect. 3. The very first question of the questionnaire always asks the user to specify the number of activities of primary interest. For this a list of available activities in log is shown to user and the user can choose the activities s/he wants to restrict from this list. Depending on the number of activities chosen different sets of questions will be triggered. For instance if the user chooses one activity of primary interest, the next question will ask about the number of times a specified activity is allowed to occur. If more than one activity (e.g., in case of our example two activities) is chosen, the questions related to relationships between chosen activities will be asked. In our example:

- Which type of limitation you would like to exert?
  - Dependent Existence; define whether the occurrence or non-occurrence of an activity imposes an obligation on occurrence or non-occurrence of another activity, e.g., define an inclusive relation between two activities.

[1] Please note that the observed behavior does not indicate a medical rule but we chose this observation to show how we can specify a behavior using *Elicit Compliance Rule* plug-in.

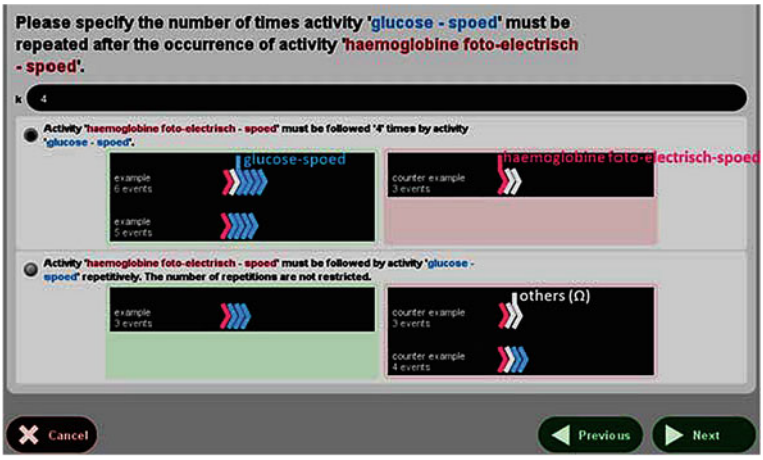

Fig. 6. Elicit compliance rule plug-in

- Bounded Existence; define whether number of occurrences of one activity is dependent to number of occurrences of the other activity.
- Sequence of Occurrence; define whether there should be a sequential relation between occurrence of two activities, e.g., define a precedence or simultaneous relation between two activities.
- Bounded Sequence of Occurrence; define whether a specified sequence must be repeated.

We choose *Bounded Sequence of Occurrence* from the list of alternative answers. As the result of this choice, a configurable pattern is selected in the back-end and questions to configure the selected pattern are presented.

The first question from the second phase will ask whether the user wants to limit the repetition of activity *'glucose-urgent'* after activity *'haemoglobin-photoelectric-urgent'* and if yes how many times *'glucose-urgent'* must occur after *'haemoglobin-photoelectric-urgent'*. Figure 6 illustrates this step in *'Elicit Compliance Rule'* plug-in in ProM where we chose: 4 times repetition of *'glucose-urgent'* after *'haemoglobin-photoelectric-urgent'*.

In order to support the user to make informed choices, for every answer a sample compliant trace and non-compliant trace is given as shown in Fig. 6. Additionally, the outcome of the currently chosen configuration is visualized to the user: the selected and partially configured rule is used to check compliance of the log w.r.t. this preliminary rule using the technique of [20]. The screen in Fig. 6 shows several compliant and non-compliant traces by which the user can use her domain knowledge to assess which answer translates her intention best.

Subsequent questions assist the user in deciding about details of the intended behavior. These questions concern configuration options which are orthogonal to each other, hence they can be resolved in any order. These questions include:

- Is it allowed that other activities occur between occurrences of activity *'haemoglobin-photoelectric-urgent'* and *'glucose-urgent'*?
- Is it allowed that other activities occur between occurrences of activity *'glucose-urgent'*?
- Is it allowed that several occurrences of activity *'haemoglobin-photoelectric-urgent'* be followed by specified repetitions of activity *'glucose-urgent'*?
- Is it allowed that activity *'glucose-urgent'* occurs before activity *'haemoglobin-photoelectric-urgent'* independently from the defined sequence?
- Is it allowed that the specified sequence of $\langle (haemoglobin\text{-}photoelectric\text{-}urgent) \underbrace{(glucose - urgent) \ldots (glucose - urgent)}_{4} \rangle$ occurs multiple times?
- Is it allowed that the specified sequence of $\langle (haemoglobin\text{-}photoelectric\text{-}urgent) \underbrace{(glucose - urgent) \ldots (glucose - urgent)}_{4} \rangle$ never occurs?
- Is it allowed that after the specified sequence $\langle (haemoglobin\text{-}photoelectric\text{-}urgent) \underbrace{(glucose - urgent) \ldots (glucose - urgent)}_{4} \rangle$, activity *'haemoglobin-photoelectric-urgent'* occurs without being followed by repetitions of *'glucose-urgent'*?

Resolving these questions yields a *configured pattern* which describes precisely the intended behavior. This Petri net can be used further for automated compliance checking.

6 Related Work

Informal description of compliance requirements can be interpreted differently in context of different business operations. Therefore precise specification of them is necessary [15]. Specification patterns are extensively used in software development [4,8,9,16,28] and also in formulating compliance requirements [10–12,27,29]. Most of these approaches use some type of structured natural language and pre-formulated templates to construct formal specifications that can then be analyzed. Often, these informal specifications are initially mapped to an intermediate representation (e.g., model-driven patterns), at which point context dependencies and ambiguities are resolved. The result is then further refined into a targeted formalism. In [10,11,29] Elgammal et al. introduce a pattern-based approach for capturing compliance requirements. Their patterns are parameterized and formalized in LTL. In order to make the approach usable for business users, they developed a tool-set where user can define compliance requirements using a specialized version of declare modeling notation. A common problem in most of above mentioned works is that pre-formulated patterns

are limited and hard coded; hence they fail to capture subtle aspects of different compliance requirements. In addition in most of the approaches, mapping and adapting patterns in a specific context requires extensive knowledge in specification languages. Our approach aims to allow compliance specification for end users without such extensive knowledge.

7 Conclusion and Future Work

The *Compliance* plug-in of ProM supports the capabilities described in this paper. The configurable compliance pattern repository is comprehensive and allows for specifying different types of compliance requirements we found in literature and many more. However, an accurate evaluation of the tool and approach is required. In future we would like to evaluate how effective the approach and tool are in practise involving business users. In the presented approach, we focused on control-flow compliance rules. We would like to investigate similar approaches for formalizing requirements restricting other perspectives of processes such as time, data, and resource. In addition we would like to check the scalability of configurable compliance patterns by applying our approach in different domains and identify compliance requirements that we are not able to specify using our current set of configurable compliance patterns.

References

1. http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisc. Rew.: Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
3. van der Aalst, W.M.P., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.H.: Configurable process models as a basis for reference modeling. In: Bussler, ChJ, Haller, A. (eds.) BPM 2005. LNCS, vol. 3812, pp. 512–518. Springer, Heidelberg (2006)
4. Abid, N., Dal Zilio, S., Le Botlan, D.: Real-time specification patterns and tools. In: Stoelinga, M., Pinger, R. (eds.) FMICS 2012. LNCS, vol. 7437, pp. 1–15. Springer, Heidelberg (2012)
5. Adriansyah, A., van Dongen, B., van der Aalst, W.M.: Conformance checking using cost-based fitness, analysis. In: EDOC'11, pp. 55–64 (2011)
6. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. J. Vis. Lang. Comput. **22**(1), 30–55 (2011)
7. Bose, R.P.J.C., van der Aalst, W.M.P.: Analysis of Patient Treatment Procedures. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 165–166. Springer, Heidelberg (2012)
8. Cobleigh, R.L., Avrunin, G.S., Clarke, L.A.: User guidance for creating precise and accessible property specifications. In: SIGSOFT FSE, pp. 208–218. ACM (2006)
9. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Property specification patterns for finite-state verification. In: FMSP. pp. 7–15. ACM (1998)

10. Elgammal, A., Türetken, O., van den Heuvel, W.J.: Using patterns for the analysis and resolution of compliance violations. Int. J. Coop. Inf. Syst. **21**(1), 31–54 (2012)
11. Elgammal, A., Turetken, O., van den Heuvel, W.-J., Papazoglou, M.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 17–31. Springer, Heidelberg (2010)
12. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and verifying workflow-based regulations. In: ReMo2V. CEUR Workshop Proceedings, vol. 241. CEUR-WS.org (2006)
13. Ghose, A.K., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
14. Governatori, G., Milosevic, Z., Sadiq, S.W.: Compliance checking between business processes and business contracts. In: Hung, P.C.K. (ed.) EDOC, pp. 221–232. IEEE Computer Society, Los Alamitos (2006)
15. Koliadis, G., Desai, N., Narendra, N.C., Ghose, A.K.: Analyst-mediated contextualization of regulatory policies. In: IEEE SCC, pp. 281–288. IEEE Computer Society (2010)
16. Konrad, S., Cheng, B.H.C.: Facilitating the construction of specification pattern-based properties. In: RE, pp. 329–338. IEEE Computer Society (2005)
17. de Leoni, M., van der Aalst, W.M.P., van Dongen, B.F.: Data- and resource-aware conformance checking of business processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 48–59. Springer, Heidelberg (2012)
18. Lu, R., Sadiq, S.K., Governatori, G.: Compliance aware business process design. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 120–131. Springer, Heidelberg (2008)
19. Smith, R.L., Avrunin, G.S., Clarke, L.A.: From natural language requirements to rigorous property specifications. In: Monterey Workshop 2003 (SEES 2003), No. UM-CS-2004-019, Chicago, IL, pp. 40–46, September 2003
20. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did I Misbehave? diagnostic information in compliance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 262–278. Springer, Heidelberg (2012)
21. Ramezani, E., Fahland, D., van Dongen, B., van der Aalst, W.: Diagnostic information in temporal compliance checking. Technical report, BPM Center Rep. BPM-12-17 (2012)
22. Ramezani Taghiabadi, E., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 304–320. Springer, Heidelberg (2013)
23. Ramezani, E., Fahland, D., van der Werf, J.M., Mattheis, P.: Separating compliance management and business process management. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part II. LNBIP, vol. 100, pp. 459–464. Springer, Heidelberg (2012)
24. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1), 1–23 (2007)
25. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
26. Sadiq, W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)

27. Schumm, D., Turetken, O., Kokash, N., Elgammal, A., Leymann, F., van den Heuvel, W.-J.: Business process compliance through reusable units of compliant processes. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 325–337. Springer, Heidelberg (2010)
28. Smith, R.L., Avrunin, G.S., Clarke, L.A., Osterweil, L.J.: Propel: an approach supporting property elucidation. In: ICSE, pp. 11–21. ACM (2002)
29. Türetken, O., Elgammal, A., van den Heuvel, W.J., Papazoglou, M.P.: Enforcing compliance on business processes through the use of patterns. In: ECIS (2011)
30. Liu, Y., Muller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Syst. J. **46**(2), 335–361 (2007)