Compliance Checking of Data-Aware and Resource-Aware Compliance Requirements

Elham Ramezani, Vladimir Gromov, Dirk Fahland, and Wil M. P. van der Aalst

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

Abstract. Compliance checking is gaining importance as today's organizations need to show that their business practices are in accordance with predefined (legal) requirements. Current compliance checking techniques are mostly focused on checking the control-flow perspective of business processes. This paper presents an approach for checking the compliance of observed process executions taking into account data, resources, and control-flow. Unlike the majority of conformance checking approaches we do not restrict the focus to the ordering of activities (i.e., control-flow). We show a collection of typical data and resource-aware compliance rules together with some domain specific rules. Moreover providing diagnostics and insight about the deviations is often neglected in current compliance checking techniques. We use control-flow and data-flow alignment to check compliance of processes and combine diagnostics obtained from both techniques to show deviations from prescribed behavior. Furthermore we also indicate the severity of observed deviations. This approach integrates with two existing approaches for control-flow and temporal compliance checking, allowing for multi-perspective diagnostic information in case of compliance violations. We have implemented our techniques and show their feasibility by checking compliance of synthetic and real life event logs with resource and data-aware compliance rules.

Keywords: compliance checking, auditing, data-aware and resource-aware compliance requirements, conformance checking

1 Introduction

Organizations need to comply with an increasing set of laws, regulations, and service level agreements set by both internal and external stakeholders. Major corporate and accounting scandals including those affecting Enron, Tyco, Adelphia, Peregrine, and WorldCom have fueled the interest in more rigorous auditing practices. Legislation, such as the Sarbanes-Oxley (SOX) Act of 2002 and the Basel II Accord of 2004, was enacted as a reaction to such scandals. Failing to comply may be costly, therefore organizations need to continuously check whether processes are executed within a given set of boundaries. Moreover organizations seek for a better control of their processes to streamline their business operation and prevent fraud, malpractices, risks, and inefficiencies.

There are two basic types of compliance checking: (1) *forward compliance checking* aims to design and implement processes where compliant behavior is enforced, and (2) *backward compliance checking* aims to detect and localize non-compliant behavior. This paper focuses on backward compliance checking based on event data.

Current backward compliance checking techniques focus on verifying aspects related to control flow while checking compliance requirements addressing other fundamental aspects of a process including data handling, and resources are as important. The compliance requirements considered in this paper take into account data, resources, control-flow, and their

interplay. We have collected several compliance requirements found in literature; we classify these requirements using two main categories and propose two generic techniques for checking these rules. Our approach seamlessly integrates with control-flow compliance checking. More important, the technique provides detailed diagnostic information in case of non-compliant behavior; it shows for each process instance *which attribute(s)* (resource or data) in *which event* violated a requirement and *what* changes would have been needed to make the behavior compliant. Our data and resource-aware compliance checking techniques leverage a recent *data-aware* alignment technique [11] that allows to check conformance of a log with respect to a *data-aware Petri net*. We adapt and improve this technique for our purpose. Moreover, our collection of data and resource-aware compliance rules, address the problem of compliance rule elicitation from informal description of compliance requirements.

The remainder of this paper is organized as follows. We recall some basic definitions for control-flow and data-flow alignments in Sect. 2. In Sec. 3, we give an overview on how a compliance requirement may impact different perspectives of a business process and show different types of data and resource-aware compliance rules. Section 4 introduces the problem of data and resource-aware compliance checking and discusses our solution through a running example. In Sect. 5 the implementation of the approach in ProM is showcased. Experimental results are presented and discussed in Sect. 6. We discuss related work in Sect. 7, and Sect. 8 concludes this paper.

2 Preliminaries

This section recalls basic notions for control-flow alignment [2,5] and data-flow alignment [12] on which we build for data and resource-aware compliance checking. Alignments relate recorded executions of a process to a formal specification which describes the boundaries of a compliant process.

Logs. Executions of business processes are recorded in terms of *event logs*. An event log L is a multiset of traces. A log trace $\sigma^L = \langle e_1, \ldots, e_n \rangle \in L$ describes a particular process execution as a sequence of events. An event e refers to the execution of an activity and is a tuple of pairs of distinct attributes and their corresponding values; $e = ((attr_1^e, val_1^e), \ldots, (attr_k^e, val_k^e))$. Each attribute and its corresponding value provides information on the executed activity. For instance in the event $e = ((name, approve\ loan\ request), (resource, John), (amount, 10000))$, the pair $(name, approve\ loan\ request)$ refers to the name of the respective activity executed, $(resource, John)$ indicates the resource approved the request, and $(amount, 10000)$ records the request amount.

Let $E = \{e_1, \ldots, e_n\}$ be the set of all events in log L. $ATTR^e = \{attr_1^e, \ldots, attr_k^e\}$ is the set of all k attributes of an event e; and $VAL^e = \{val_1^e, \ldots, val_k^e\}$ is the set of all values for attributes of an event e. As a shorthand, we write $l^e(attr_i^e) = val_i^e$ to denote that event e has value val_i^e for attribute $attr_i^e$, $1 \leq i \leq k$. We define $ATTR^L$ as the set of all event attributes in the log; $ATTR^L = \cup_{e \in E} ATTR^e$.

Specified behaviors. A specification describes what is considered compliant and what is not. In essence, each specification describes a set S of *compliant traces*. Every compliant trace $\sigma^S = \langle a_1, \ldots, a_n \rangle \in S$ is a sequence of activities which are allowed to be executed. Activities may have attributes with corresponding admissible values that describe how an activity must be executed in a compliant trace. An activity a is a tuple of pairs of distinct attributes and their corresponding *set* of admissible values; $a = ((attr_1^a, Val_1^a), \ldots, (attr_k^a, Val_k^a))$ where each Val_i^a is a set of admissible values that attribute $attr_i^a$ is allowed to take.

For instance for the activity $a = ((name, \{approve\ loan\ request\}), (resource, \{John, Elham, Luis\}), (amount, \{500, \ldots, 1500\}))$, the pair $(name, \{approve\}\ loan\ request\})$

shows the name of the activity allowed to be executed, $(resource, \{John, Elham, Luis\})$ indicates the resources who are allowed to approve a loan request, $(amount, \{500, \ldots, 1500\})$ specifies the admissible amount for a loan.

As a shorthand, we write $l^a(attr_i^a) = Val_i^a$ to denote that activity a can take values Val_i^a for attribute $attr_i^a$, $1 \leq i \leq k$. The set $ATTR^a$ of all attributes of activity a, and $ATTR^S$ of specification S are defined similar to attributes of events and logs.

Given an activity a, l^a assigns a set of admissible values to an attribute of activity a; $l^a(attr_i^a) = Val_i^a$ for $1 \leqslant i \leqslant k$. We define $ATTR^a$ as the set of all k attributes of an activity a; $ATTR^a = \{attr_1^a, \ldots, attr_k^a\}$. $ATTR^S$ is the set of all attributes of all the activities in the specification; $ATTR^S = \cup_{a \in A} ATTR^a$, and A is the set of all activities in specification S; $A = \{a_1, \ldots, a_n\}$.

Relating events in a log to activities in the specification. As we already mentioned every event refers to execution of an activity. We will explain later in this paper how we use alignment techniques for checking compliance. There, we will pair events in the log and activities in the specification using the following notation.

For event log L with events E, and attributes $ATTR^L$, we call an attribute $type^L \in ATTR^L$ a *log type attribute* if $type^L \in \cap_{e \in E} ATTR^e$, i.e., an attribute present in all events in L. We assume that at least one $type^L$ exists for L. We define $TYPEVAL^L$ as the set of all values for $type^L$; $TYPEVAL^L = \{l^e(type^L) | e \in E\}$.

Similarly for specification S with activities A and attributes $ATTR^S$, we call an attribute $type^S \in ATTR^S$ a *specification type attribute* if $type^S \in \cap_{a \in A} ATTR^a$, i.e., an attribute present in all activities of specification S. We assume that at least one $type^S$ exists for specification S and attribute $type^S$ has only a single value for each activity i.e., $l^a(type^S) = \{label^a\}$. We define $TYPEVAL^S$ as the set of all admissible values for a specification attribute $type^S$; $TYPEVAL^S = \cup_{a \in A} l^a(type^s)$.

Note that a log and a specification may have multiple type attributes. Later on, we relate L to S by picking a specific log type attribute and a specific specification type attribute each, and mapping the values $TYPEVAL^L$ to the values $TYPEVAL^S$.

Data-aware Petri nets The specification S representing all compliant traces, can be expressed in various ways. For instance as a Petri net [2] or in terms of declarative constraints [12]. Typically, the specification S is very large, being the semantic notion of all compliant traces. In this paper, we use Petri nets to specify S in a concise form. Fig. 1 illustrates a variant of Petri nets called *data-aware* Petri net. Data-aware Petri nets extend classical Petri nets with data. They describe which activities are allowed to be executed and in which sequence. In addition, a data-aware Petri net describes how every activity is allowed to be executed with respect to its attributes and their values. N^S —the data-aware Petri net shown in Fig. 1— describes a simplified version of a loan application procedure. The process starts with activity a_1 which is represented with the transition *receive loan request* and continues with a_2 represented by *check credit* of the applicant requesting the loan. Afterwards the decision for *approving* or *rejecting* the request should be taken either by executing activity a_3 or a_4, and finally the applicant will be *informed* about the decision by executing activity a_5. Each activity has an attribute *name* that we will use later as a specification attribute $type^S$ for mapping (please see Sect.2). The admissible value of this attribute is written inside the transition representing each activity.

Moreover N^S depicts other admissible attributes as the yellow colored eclipses including *resource*, and *amount*. Some activity attributes may be accessible by all the activities e.g., *resource* or may be accessible by specific activities e.g., *amount*. This is specified by the *dashed line* connecting attributes to activities. A specification describes how an activity must

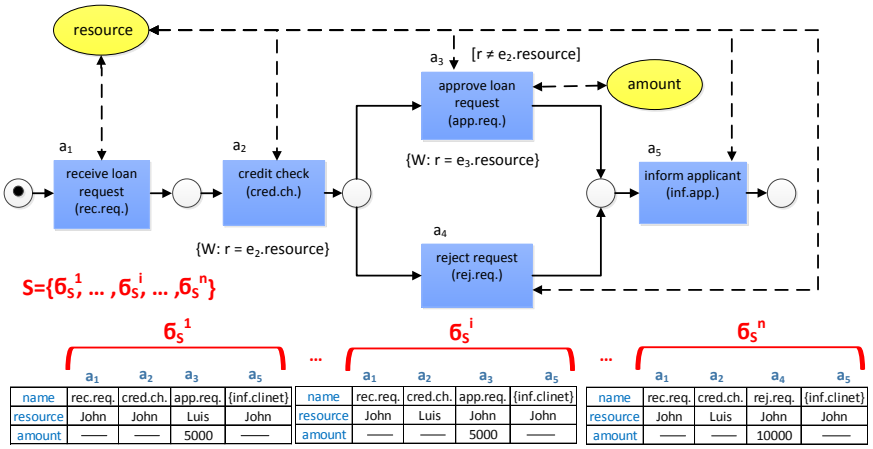

Fig. 1. N^S, a data-aware Petri net example, and some compliant trace examples from specification S.

be executed in order to be compliant; hence in a data-aware Petri net activities may have additional annotations. For instance in N^S, activity a_3 represented by the labeled transition *approve loan request* is guarded. The guard restricts the relation between activities of a_2, and a_3 with the help of variable r. The value of the attribute *resource* for the recent event produced by the execution of a_2 is stored in variable r. This is expressed by the *write statement* $\{r = e_2.resource\}$ annotated at a_2. The Execution of activity a_3 updates the value of r. This is expressed by the *write statement* $\{r = e_3.resource\}$ annotated at a_3. The guard on activity a_3 $[r \neq e_2.resource]$ specifies that the *resource* executed a_3 must be different with the *resource* executed a_2. That is, the new value of r must be different from its previous value which was written by the execution of a_2. Please note that this guard describes *four-eye principle*. The firing sequences of the Petri net N^S is the set S given in Fig. 1(bottom).

Aligning observed behavior to specified behavior. An observed trace in a log may deviate from admissible behaviors; e.g., the non-compliant trace σ^L shown in Fig. 2. There are three events and three attributes in this trace. We choose *name* as log type attribute $type^L$ (see Sect.2) and map the values of this attribute occurring in σ^L to transition labels in N^S when they have the same value. Please note that transition labels in N^S denote the admissible values of $type^S$. If we only consider the compliance of σ^L w.r.t. existence and correct sequence of events, we see that an event *e* with (*name:inform applicant*) is missing as the final event in the trace. To understand where σ^L deviates from specification S, we apply *control-flow alignment* [2] between σ^L and N^S as follows.

	name	rec.req.	cred.ch.	app.req.
L	resource	John	Luis	Luis
	amount	——	——	10000

Fig. 2. Sample trace σ^L

The idea is to find a compliant trace $\sigma^S \in S$ that is as similar as possible to σ^L; differences between σ^S and σ^L then indicates deviations. We relate σ^L to any trace $\sigma^S \in S$ by pairing events in σ^L to activities in S where events and activities are of the same type with the same value. For our example $type^L = type^S = name$. The obtained alignment is shown in Fig. 3; top row corresponds to events in σ^L and bottom row refers to activities in σ^S. As is indicated in

L	name	rec.req.	cred.ch.	app.req.	>>
	resource	John	Luis	Luis	
	amount	——	——	10000	
S	name	rec.req.	cred.ch.	app.req.	inf.client

Fig. 3. A control-flow alignment γ_c of trace σ^L and specification S with $type^L = type^S = name$ and *mapping*: $R(val) = val$.

γ^c, an event with (*name:inform client*) is missing; this is denoted by $\gg$ in the alignment indicating a move on model.

Let L be an event log and S be a specification. We pick a log type attribute $type^L$ (with values $TYPEVAL^L$ occurring in L) and a specification type attribute $type^S$ (with values $TYPEVAL^S$ occurring in S) as described above. Further, we pick a *mapping* R : $TYPEVAL^L$ to $TYPEVAL^S$ relating log type attributes to specification type attributes. A control-flow alignment move m^c (wrt. $type^L$, $type^S$, and R) is a pair $(x, y) \in (E \cup \{\gg\}) \times (A \cup \{\gg\}) \backslash \{(\gg, \gg)\}$ where:

- $(x, \gg)$ is a *move on log*.
- $(\gg, y)$ is a *move on specification* S.
- (x, y) is a *synchronous move* if $x, y \neq \gg$ and $R(l^x(type^L)) \in l^y(type^S)$.

A synchronous move (x, y) relates the event x to the activity y based on the log type attribute $type^L$ and the specification type attribute $type^S$; the type value of x has to map to the type value of y (via mapping R).

A control-flow alignment of a trace $\sigma^L \in L$ to S is a sequence $\gamma_c = \langle m_1^c, \ldots, m_n^c \rangle$ of control-flow alignment moves such that ignoring $\gg$, the projection $x_1 \ldots x_n|_E$ is the original trace σ^L, and the projection $y_1 \ldots y_n|_A = \sigma^S \in S$ is described by the specification.

The control-flow conformance checking technique of [2,5,14] returns an *optimal* control-flow alignment s.t. no other alignment has fewer non-synchronous moves (move on log or move on specification only). This technique finds an optimal alignment using a cost-based approach: a cost function κ_c assigns each control-flow alignment move (x, y) a cost $\kappa_c(x, y)$ s.t. a synchronous control-flow alignment move has cost 0 and all other types of moves have cost> 0. Then an A^*-based search on the space of (all prefixes of) all alignments of σ^L to S is guaranteed to return an optimal alignment for σ^L and S. In such an optimal alignment, a move on log $(e, \gg)$ indicates that the trace σ^L had an event e that was not supposed to happen according to the specification S whereas a move on specification $(\gg, a)$ indicates that σ^L was missing an event that was expected according to S. As the alignment preserves the position relative to the trace σ^L , we can locate exactly where σ^L had an event too much or missed an event compared to S.

A data-aware alignment [11] extends a control-flow alinement by also comparing every event $e \in \sigma^L$ with all its attributes and their values to its corresponding activity $a \in \sigma^S$ with its admissible attributes and their values. If an event e was not executed in compliance with its corresponding activity, the data-aware alignment technique can identify the deviation and extent of the deviation.

Given event log L and specification S, a control-flow alignment $\gamma_c = \langle m_1, \ldots, m_n \rangle$ of a log trace $\sigma^L \in L$ to S extends to a data-aware alignment $\gamma_d = \langle m_1, \ldots, m_n \rangle$ by considering all attributes as follows. Log move and model move are defined as before. A synchronous move (e, a) is *correct* iff for each attribute $attr_i \in ATTR^e$ holds: $attr_i \in ATTR^a$ and $l^e(attr_i) \in l^a(attr_i)$, i.e., each data attribute of the event has a value admitted by the activity. Otherwise, the synchronous move is called *incorrect*. Figure 4 shows an example of a data-aware alignment.

L	name	rec.req.	cred.ch.	app.req.	>>
	resource	John	Luis	**Luis**	
	amount	—	—	10000	
S	name	rec.req.	cred.ch.	app.req.	inf.client
	resource	John	Luis	John	John
	amount	—	—	10000	—

Fig. 4. The data-aware alignment γ_d of trace σ^L and σ^S.

Activity a_3 named *approve loan request* is executed by a *resource* which is not allowed based on the guard at a_3 in N^S. The resource (*Luis*) executed *approve loan request* should

have been different from the resource executed *credit check*. In addition to finding deviations, γ_d indicates the correct value for the violating attribute. In our example, from the resources allowed to execute activity *approve loan request*, *John* is suggested as the correct value.

Similar to the cost-based approach applied in control-flow alignment, a cost function κ_d assigns each data-alignment move a cost $\kappa_d(x, y)$ s.t, a correct data-alignment move has cost 0 and an incorrect move has cost> 0. In the data-aware alignment technique of [11], an ILP solver finds among all synchronous control-flow alignment moves, values for attributes of S such that the total cost of deviations for an alignment including *move on log*, *move on model*, and *incorrect move* is minimized. We apply control-flow and data-aware alignments for data and resource-aware compliance checking.

3 Compliance Requirements

Compliance requirements prescribe how internal or cross-organizational business processes have to be designed or executed. They originate from legislations and restrict one or several perspectives of a process (control flow, data flow, process time or organizational aspects). Restrictions can be imposed for individual cases or groups of cases, they can prescribe properties of process executions or process design [14]. These different aspects of compliance give rise to the framework shown in Fig. 5. A complex compliance requirement covering several perspectives of a process can be decomposed into smaller compliance rules, each covering a single aspect along the dimensions of this framework.

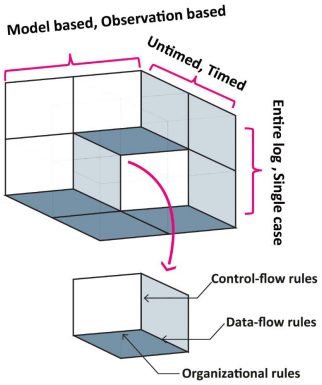

Fig. 5. Compliance Rule Framework [14]

For example, a compliance requirement might state: "Before approving a loan request, the bank must check the credit history of the applicant. A request can be approved if it passes the evaluation. The approval and credit checking must be done by different agents. Regardless of rejection or approval of the request, applicant must be informed about the bank's decision within a weak after submitting the request".

This requirement can be divided into different compliance rules: *i) control flow*: 'Loan request approval or rejection must be preceded by credit check', *ii) process data*: 'A loan may be approved only if it passes the evaluation', *iii) process resource*: 'Request approval and check credit must be done with different resources', and *iv) process time*: 'Every application for requesting a loan must be processed within one week from the date of application submission'; each rule taking only one perspective into account. Our earlier compliance checking techniques of [14,20] are able to check control-flow and temporal compliance rules, but do not provide a notion of *data* or *resource*. In the following, we present a new approach for checking data and resource-aware compliance rules that provides diagnostic information about deviations. Before, we will explain the data and resource-aware rules supported by our approach.

3.1 Data-Aware and Resource-Aware Compliance Rules

We identified some works [22,4,18,7,21,8,19] discussing compliance rules that restrict process data and resource. Table 1 shows the collection of compliance rules taken from these sources and some more taken from practise e.g., medical guidelines. We found some typical

restrictions on process data and resources such as four-eye principle (separation of duties), authorization level or three-way match and some domain specific compliance rules.

In addition to the classification presented in Table 1, all different types of data-aware and resource-aware constraints fall into two main categories: (1) constraints that enforce a restriction on data attributes, and (2) constraints that restrict activities when a certain data condition holds. For example a compliance rule such as the *four-eye principle* is of the first category. This rule specifies that two activities A and B must be executed by two different resources. The rule assumes that the underlying control-flow sequence is correct i.e., no matter in which order two activities A or B are executed, the restriction is on the corresponding data attribute (resource). In case A is executed first by resource R_1, the rule will urge that B must not be executed by R_1. Whereas, if B was executed first, the restriction would be on the resource executing activity A. In contrary with rules like *four-eyes principle*, a rule stating "Activity B must not be executed for *gold* customers" belongs to the second category. This rule restricts the execution of activity B when a certain value ($gold$) for data attribute *customer type* holds. That is, based on a certain data condition, the constraint restricts the control-flow i.e., restricting execution of activity B.

In Sect. 2 we discussed an example of a rule from the first category. In the next section we will discuss examples of both categories in more detail while elaborating on the differences in corresponding checking technique employed.

Rule Description	**Example**
Four-eye principle: A security principle that requires segregating the execution of critical tasks and associated privileges among multiple users.	The person requesting purchase of goods should not be the one who approves it.
	A purchase order approval requires two signatures.
Authorization (Access control): A security principle that limits execution of activities or accessing a data object to authorized individuals.	Only a financial manager can approve a loan.
Two (three)-way match :An accounting rule that requires the value of two different data objects to match.	All vendor invoices that are based on purchase orders should be matched with purchase order lines (two-way matching).
Activity T may/must (not) be executed if attribute X has the value v; (X may be local to the activity T or may appear anywhere in a trace).	An account must not be opened in case risk is high.
	During ventilation, patient must receive "propofol" with dosage of (5mg).
Activity T_1 may/must (not) be executed if attribute X has value v at activity T_2. (attribute X is local to activity T_2)	In case the respondent bank rating review is rejected during evaluation, an account must never be opened.
Activity T must not change value of attribute X.	Bank account data must not change during payment.
Value of attribute X must not change after activity T is executed.	All invoices must be archived and no changes must be made to the document.
Activity T_1 may occur only if the value of attribute X is increased/decreased by activity T_2 with d.	If gastric tube feeding cannot be increased by (1,20 kcal/ml), then use 'Erythromycin'.
If attribute X has value v then resource R must execute the activity.	Loans with value more than 1000000 Euro must only be approved by CFO.
If activity T_1 is done by agent A, activity T_2 must be done by the same agent.	A customer complain must be handseled with the same agent registered the customer request.

Table 1. Collection of data-aware and resource-aware compliance rules.

4 Data-Aware and Resource-Aware Compliance Checking

This section presents our main contribution, an approach for checking data and resource-aware compliance on past executions recorded in event logs. We first introduce a motivating example which we use throughout this section to explain our techniques.

4.1 Motivating Example

A process model, expressed in BPMN notation, describing a simplified procedure for procurement in a company is shown in Fig. 6. The process starts with activity *receive request for purchasing goods*. Afterwards the sub-process for *choosing supplier* is activated. Every purchase requisition carries a risk which is calculated based on a set of factors such as history of supplier's business, legal background, and etc. Consequently the risk is classified as *high* or *low*. If risk is high and not acceptable, the procurement expert investigates about *risk reduction measures*. These measures may lead to a *low* risk or the risk may stay still *high*. The procurement expert must decide based on the new information to continue with *risk reduction measures* or *accept the risk*. If risk is *accepted purchase order* can be *prepared*. For every *purchase order* two *approvals* are required by two different agents. An approved purchase order is *sent to supplier*. The process continues with *receiving invoice*. After *receiving goods*, the *payment* is done and the process terminates.

To prevent fraud, the company has defined various compliance rules. The employees may deviate from the modelled process as long as they do not violate the compliance rules. Here, we present just two of the rules that must be followed:

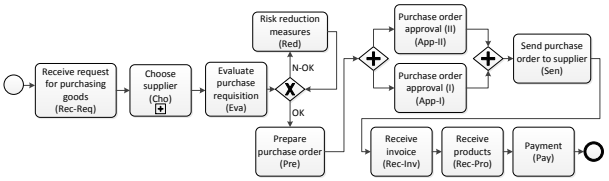

Fig. 6. A process model for procurement

- *Rule 1* (Four-eye principle): *purchase order approval(I)* and *purchase order approval(II)* must be executed by two different agents.
- *Rule 2*: If *risk* of procurement from a supplier is calculated *high*, *risk reduction measures* must be executed at least once.

During auditing, the company checks if the procurement process were executed in compliance with relevant compliance rules including above mentioned ones. Therefore, executions of this process are recorded in terms of an event log and are used for compliance checking. These rules cannot be checked by solely considering the execution order of events; instead, specific event attribute values need to be taken into account.

L	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
	resource	John	Luis	Luis	Sara	Arash	Clara	Luis	Luis	John	Luis
	role	clerk	expert	expert	clerk	director	manager	expert	expert	clerk	expert
	inventory level	500 pce.	300 pce.	—	—	—	—	—	—	—	—
	risk	—	low	high	high	high	high	high	high	high	high
	time	20-Jan-14	21-Jan-14	22-Jan-14	24-Jan-14	26-Jan-14	26-Jan-14	28-Jan-14	28-Jan-14	29-Jan-14	29-Jan-14

Fig. 7. Trace σ recorded an instance of the procurement process execution

The synthetic event log L contains the executions of the procurement process for January of 2014. The trace σ shown in Fig. 7 is taken from the event log L.

4.2 Methodology

As discussed in 3, compliance rules fall into two main categories. The techniques we employ to check compliance of rules varies per category. Figure 8 depicts the approach we use for checking the compliance rules of the first category. For these rules, we need to check if a restricted activity was executed with a correct data or resource. Hence, the restriction on data attributes must be checked and the underlying control-flow is assumed to be correct. For this, as is shown in Fig 8, we prepare the log and enrich it with some necessary information in *steps 1,* and *2*. To check every compliance rule, we need to capture the scope of the rule i.e., we identify when a compliance rule is triggered. The scope of the rule is defined based on the occurrence of an activity or sequence of activities creating a control-flow alignment. In *step 3*, we specify the scope of the compliance rule in form of a classical Petri net and create a control-flow alignment in *step 4*. We generate a log using the alignment result and enrich it in *step 5* with diagnostics we obtained in the previous step. Later in *step 6*, we create a data-aware alignment using a data-aware Petri net to check if the data condition specified in the rule, holds.

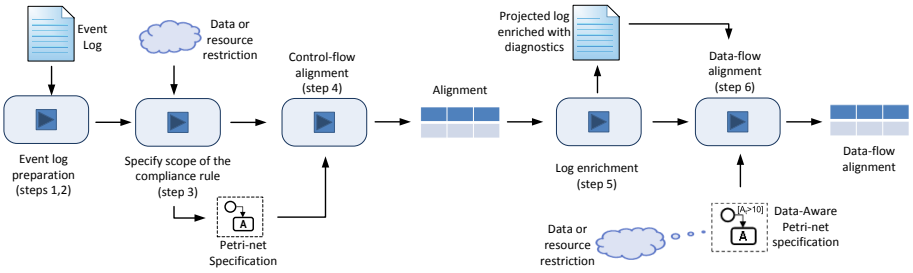

Fig. 8. Compliance checking approach for compliance rules restricting a data attribute

The compliance rules of the second category restrict the execution of activities when a certain data or resource condition holds. These rules assume the data-flow is correct and the control-flow i.e., the execution of activities under a certain data condition, must be checked. For this, we create a data-aware alignment to identify all the situations where a compliance rule must hold. Then creating a control-flow alignment we can check if activities were executed correctly under the specified data condition. Fig. 9 describes the approach we employ for checking compliance rules of the second category. In this approach after preparation of the event log in *steps 1, 2,* and *3*, in *step 4* we apply data-aware alignment technique to identify all the situations where the compliance rule must hold. In *step 5* we enrich the log with diagnostics obtained in data-aware alignment. We define the scope of the compliance rule in *step 6* and apply control-flow alignment technique in *step 7* to check if

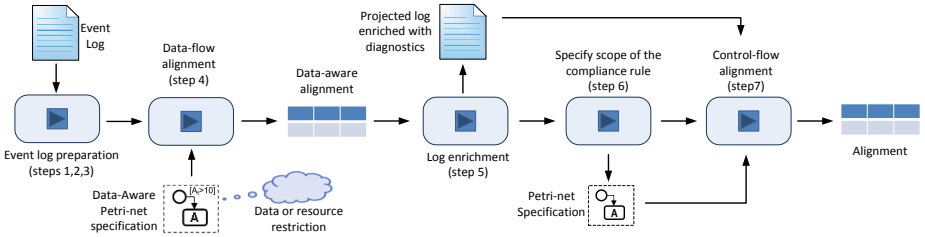

Fig. 9. Compliance checking approach for compliance rules restricting activities when a certain data or resource condition holds

activities were executed correctly under the specified data condition. In the following we will elaborate on both approaches by checking the sample compliance rules defined for the motivating example.

4.3 Compliance Checking of Rules Restricting Data Attributes

From the compliance rules listed for the motivating example in Sect. 4.1, *Rule 1* falls in the first category of compliance rules. For checking such rules we employ the approach shown in Fig. 8. In the following we will explain the technique step by step for this example.

Compliance Rule 1 (four-eye principle): Purchase order approval(I) and purchase order approval(II) must be executed by two different agents.

Step 1. In this step we abstract the log from event attributes and their values which are not relevant for the rule. This rule restricts the data attribute *resource* of activities with (*name: App-I*), and (*name: App-II*). Therefore we can focus on event attributes *name*, and *resource* and discard other event attributes of σ. Consequently we obtain σ_{11} shown in Fig. 10.

L	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
	resource	John	Luis	Luis	Sara	Arash	Clara	Luis	Luis	John	Luis

Fig. 10. Trace σ_{11} obtained from *step 1*

Step 2. In the second step we abstract σ_{11} further from information that is not relevant for checking *Rule 1*. Therefore we keep the value of data attribute *name* when *(name: App-I)*, and *(name: App-II)* and replace all other values of attribute *name* with a generic value Ω. *Rule 1* restricts the value of data attribute *resource* at activities with *(name: App-I)* and *(name: App-II)*; hence we keep the value of *resource* at these activities and discard its value at any other event. As a result of this step we obtain the trace σ_{12} shown in Fig. 11.

Step 3. The four-eye rule belongs to the first category (Fig. 8). Hence, the control-flow is assumed to be correct. For this rule activities with *(name: App-I)*, *(name: App-II)* may be executed in any order. Considering this information, in this step we decide where the compliance rule is triggered. As soon as any of the events with *(name: App-I)* or *(name: App-II)* occurs, the rule is activated.

L	name	Ω	Ω	Ω	Ω	App-I	App-II	Ω	Ω	Ω	Ω
	resource	–	–	–	–	Arash	Clara	–	–	–	–

Fig. 11. Trace σ_{12} obtained from *step 2*

Some compliance rules may hold several times in a trace. To elaborate more on this, consider the compliance rule stating : *"every time activity **A** occurs it must be followed by activity **B**."*. If **A** occurs multiple times in a trace, we will have different instances of this rule in a trace and every occurrence of **A** must be followed by activity **B**. When deciding on the scope of a compliance rule, we need to consider if multiple instances of a compliance rule is allowed or not.

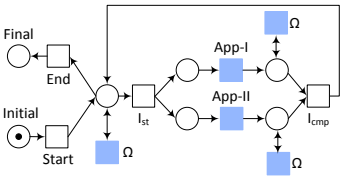

Fig. 12. Petri net specifying the scope of *Rule 1*

After defining the rule scope, we design a Petri net which describes the rule. For this, we can apply the *compliance elicitation* technique available in [15]. The Petri net describing *Rule 1* is shown in Fig. 12. This net starts by firing transition $Start$ and a token in place $Final$ represents a completed trace. The part between transitions I_{st} and I_{cmp} represents an instance of the compliance rule. Please note that

several instances of the compliance rule are allowed and captured in the structure of the net. The rule becomes active when I_{st} fires and it ends when I_{cmp} fires. After the instantiation of every compliance rule, activities with *(name: App-I)*, and *(name: App-II)* may occur in any order. The Ω transition represents any event in the trace with *name:*Ω. The hollow transitions ($Start, I_{st}, I_{cmp}$, and End) are invisible.

Step 4: Having trace σ_{12} and the Petri net, we apply our control-flow alignment technique to capture the scope where the compliance rule must hold. This alignment indicates deviations of σ_{12} from the Petri net and diagnostics about the deviations. For control-flow compliance checking we ignore attributes other than *name* and use this attribute for mapping events and activities in aligning trace σ_{12} to the Petri net. γ_1^c in Fig. 13 shows the resulting alignment.

L	name	>>	Ω	Ω	Ω	Ω	>>	App-I	App-II	>>	Ω	Ω	Ω	Ω	>>
	resource		–	–	–	–		Arash	Clara		–	–	–	–	
S	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End

Fig. 13. Alignment γ_1^c obtained from step 4.

The alignment shows that trace σ_{12} can be replayed without any real violation on the Petri net. Please note that the missing events ($\gg$) indicated in the top row of the alignment are related to invisible transitions in the Petri net and are not considered as real violations i.e., costs of deviation for these events are 0.

L	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End
	resource	—	—	—	—	—	—	Arash	Clara	—	—	—	—	—	—

Fig. 14. Trace σ_{15} enriched with additional information obtained from control-flow alignment.

Step 5: In this step we enrich trace σ_{12} with the additional information we obtained from the control-flow alignment as follows. We insert artificial events with attribute *name* and values ($Start, I_{st}, I_{cmp}, End$) in the trace where the alignment identified missing events ($\gg$). As a result we obtain trace σ_{15} shown in Fig. 14. This additional information marks the scope of the compliance rule in the trace and it shows that *Rule 1* was triggered once in the trace.

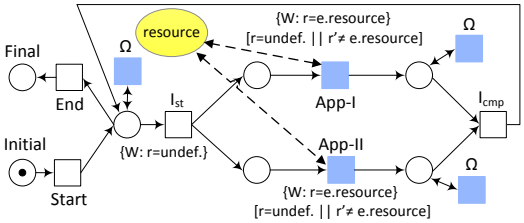

Fig. 15. Data-aware Petri net of $Rule_1$

Step 6: In this step we apply the data-aware alignment technique on the enriched log (σ_{15}) obtained from step 5 and the extended Petri net with data annotations. Figure 15 shows the data-aware Petri-net describing the restriction on data attribute *resource* w.r.t. *Rule 1*.

The data attribute *resource* is shown as the yellow eclipse in Fig. 15. The dotted line which associates attribute *resource* to transitions *App-I*, *App-II* indicates that the value of attribute *resource* can get updated by occurrence of any of these activities. As explained in Sect. 2, we define a variable R that takes the value of attribute *resource* whenever this attribute is updated by any of the transitions *App-I* or *App-II*. This is shown by the annotation $\{W : r = e.resource\}$ on these two transitions. Please note that we annotate the transition I_{st} with $\{W : r = undef.\}$ to express that as soon as the compliance rule is activated the value of variable r is set to *undefined*. The restriction on data attribute *resource* which is specified in *Rule 1* is expressed by the guard $[r = undef.||r' \neq e.resource]$ annotated at transition *App-I* and *App-II*. This guard specifies that the variable r is allowed to get a new value if its previous value is *undefined* $[r = undef.]$ or in case it has already a value, the new value must be different with current value $[r' \neq e.resource.]$; implying activities *App-I* and

App-II must be executed with different *resources*. The data-aware alignment of σ_{15} against the data-aware Petri net is shown in Fig. 16. The alignment illustrates that both guards are evaluated to correct. Hence the trace is compliant with *Rule 1*.

4.4 Compliance Checking of Rules Restricting Activities When a Certain Data Condition Holds

The compliance *Rule 2* listed for the motivating example in Sect. 4.1 falls in the second category of compliance rules. For checking these rules we employ the approach shown in Fig. 9. Next we will explain this technique step by step for *Rule 2*.

L	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End
	resource	—	–	–	–	–	–	Arash	Clara	–	–	–	–	–	–
S	name	Start	Ω	Ω	Ω	Ω	I-st	App-I	App-II	I-cmp	Ω	Ω	Ω	Ω	End
	resource	–	–	–	–	–	–	Arash	Clara	–	–	–	–	–	–

Fig. 16. Data-aware alignment γ_1^d obtained from step 6.

Compliance Rule 2: If risk of procurement from a supplier is calculated high, risk reduction measures must be executed at least once.

Step 1. Similar to the log preparation procedure we followed in the previous approach, we first abstract the log from event attributes and their values which are not relevant for the rule. Therefore we remove other attributes apart from *name* and *risk* from the trace and we obtain σ_{21} in Fig. 17.

L	name	Rec-Req	Cho	Eva	Pre	App-I	App-II	Sen	Rec-Inv	Rec-Pro	Pay
	risk	—	low	high	high	high	high	high	high	high	high

Fig. 17. Trace σ_{21} obtained from *step 1*

L	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
	risk	–	low	high	high	high	high	high	high	high	high
	data condition	Ω	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write

σ_{22} σ_{23}

Fig. 18. Traces σ_{22}, and σ_{23} obtained respectively from *steps 1, and 2*

Step 2. We keep the value of event attribute *name* for *risk reduction measure* and substitute all other values of attribute *name* with the generic value Ω. In addition we keep the attribute *risk* with all its values. Trace σ_{22} shown in Fig. 18 is the result of step 2. As is shown in σ_{22}, activity *risk reduction measure* was never executed.

Step 3. In *Rule 2*, the restriction is on the number of occurrences of activity *risk reduction measures* when data attribute *risk* has value *high*, hence the rule is triggered as soon as attribute *risk* gets value *high*. Not all the events have access to attribute *risk* i.e., not all events can *update* the value of attribute *risk*. Therefore, we need to capture the existence and changes in the value of *risk*. To this end we introduce a new event attribute named *data condition* which gets the value *risk write* whenever an event records a value for its attribute *risk*. The *data condition* gets the value Ω if *risk* doesn't have any value in an event. Trace σ_{23} shown in Fig. 18 is obtained from step 3.

Step 4. As discussed in Sect. 2, we express data and resource-aware compliance rules using a data-aware Petri net. The corresponding alignments of a data-aware Petri net to a log will then allow to check these rules. For the alignment we use the abstracted and enriched log σ_{23} obtained from *step 3*.

The data-aware Petri net shown in Fig. 19 has two transitions Ω and *risk write* where we map them respectively to the values of event attribute *data condition* in the log. Therefore, events with (*data condition:risk write*) are mapped to transition *risk write* in the data aware Petri net and events with (*data condition:*Ω) are mapped to the transition labelled Ω. The other event attributes *name* and *risk* are mapped respectively to data attributes *name* and *risk*.

We would like to capture all situations where the data condition of *Rule 2* holds. That is, we want to capture the events where *risk* is *high* and check if such an event is followed at least once by the activity *risk reduction measure*. Hence the transition *risk write* in the data-aware Petri net (Fig. 19) is guarded with *[e.risk = high]*.

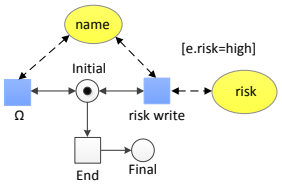

Fig. 19. The data-aware Petri net of *Rule 2*

We check the conformance of the trace σ_{23} against the data-aware Petri net of *Rule 2* applying data-aware alignment. The data-aware alignment technique checks if events with *data condition:risk write* were executed with the admissible value (*high*) for its *risk* attribute. As a result the data-aware conformance checking technique will return a data-aware alignment γ_2^d as is shown in Fig. 20. The columns colored in red indicate the events with *(data condition:risk write)* which didn't have the value *high* for attribute *risk* (yellow eclipses in the data-aware Petri net).

L	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
	risk		low	high	high	high	high	high	high	high	high
	data condition	Ω	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write
S	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
	risk		high	high	high	high	high	high	high	high	high
	data condition	Ω	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write

Fig. 20. Data alignment γ_2^d of trace σ_{23} for compliance *Rule 2*

Step 5. In this step we enrich trace σ_{23} with diagnostics we obtained from the data-aware alignment in the previous step as follows:

i) We insert an event attribute *condition holds* to all events in the trace. This new event attribute gets no value ($\gg$) if no value is recorded for the attribute *risk* of the events. Whenever a deviation is indicated in the alignment (columns colored in red), *condition holds* gets value *false*, else it gets the value *true*. Note that value *false* for attribute *condition holds* indicates all the situations in the trace where *risk* was not *high*.

ii) We insert event attribute *admissible risk* to the violating events. This new attribute shows the admissible value; for our example (*admissible risk:high)*.

iii) We insert event attribute *combined name&condition* to all events. The value of this attribute for every event is the combination of the value for the corresponding attributes *name* and *condition holds* of that event. This event attribute is added to enable us in the next step to check if an activity was executed with the right data or resource.

As a result of above mentioned modifications, we obtain trace σ_{24} (shown in Fig. 21) enriched with diagnostics and added information.

Step 6. In this step we define the scope of the compliance rule. *Rule 2* is triggered whenever *risk* is calculated *high* and it is satisfied as soon as activity *risk reduction measure* is executed. Please note that afterwards the *risk* may stay high or *risk reduction measure* may occur arbitrary number of times. If we interpret this rule in context of the trace σ_{24}, we can rephrase

L	name	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω
	risk		low	high	high	high	high	high	high	high	high
	data condition	Ω	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write
	condition holds	>>	false	true	true	true	true	true	true	true	true
	admissible risk	>>	high	high	high	high	high	high	high	high	high
	combined name& condition	Ω->>	Ω-false	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true

Fig. 21. Trace σ_{24} enriched with diagnostics obtained from data-aware alignment

the rule to: "occurrence of an event with *(combined name&condition:*$\Omega - true$) must be followed at least once by an event having value *risk reduction measures-true* or *risk reduction measures-false* for event attribute *combined name&condition*. In other words, an event with $\Omega - true$ indicates that first activity which calculated *(risk:high)* is executed. In addition any of the events with *risk reduction measures-true* or *risk reduction measures-false* indicates that activity *risk reduction measures* was executed.

When defining the scope of *Rule 2*, we need to consider whether multiple instances of the rule are allowed. In case of *Rule 2*, only fulfilling one instance of the rule is enough and after fulfilling it once, it won't be triggered again although *risk* may still be *high*. After deciding about the scope of the rule we can design a Petri net which describes the rule. The result is shown in Fig. 22.

Step 7. Having trace σ_{24} and the Petri net specification of in Fig. 22, we apply control-flow conformance checking to get an alignment between σ_{24} and the specification. This alignment indicates deviations and diagnostics about the deviations. For control-flow alignment we ignore attributes other than *'combined name&condition'* and align the trace σ_{24} to the net shown in Fig. 22. γ_2^c in Fig. 23 shows the alignment obtained.

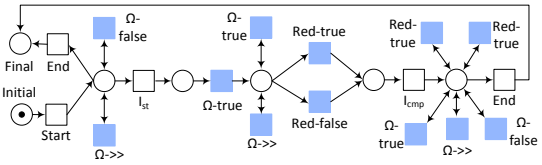

Fig. 22. Petri net specifying compliance *Rule 2*

L	name		Ω	Ω		Ω	Ω	Ω	Ω	Ω	Ω	Ω	Ω				
	risk			low		high	high	high	high	high	high	high	high				
	data condition		Ω	risk write		risk write	risk write	risk write	risk write	risk write	risk write	risk write	risk write				
	condition holds	>>	>>	false	>>	true	true	true	true	true	true	true	true	>>	>>	>>	
	admissible risk		>>	high		high	high	high	high	high	high	high	high				
	combined name& condition		Ω->>	Ω-false		Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true				
S	combined name& condition	Start	Ω->>	Ω-false	I-st	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Ω-true	Red-true	I-cmp	End	

Fig. 23. Data-aware alignment γ_2^c obtained from step 7.

The alignment indicates a deviation in trace σ_{24} implying that *risk* has been *high*, hence activity *risk reduction measure* should have happened but it didn't.

5 Implementation

The presented technique is supported by two novel plugins in the *ComplianceFramework* package of ProM 6.3 [1]. The *Data and Resource Compliance Checking-Rules Restricting Activities* plug-in checks the compliance of event logs against rules of the first category and takes an event log and a data-aware Petri net as input. The *Data and Resource Compliance Checking-Rules Restricting Attributes* checks the compliance of event logs against rules of the second category and takes an event log, a data-aware Petri net, and a classical Petri net as input. Both return diagnostics about deviations in the from of an alignment. The resulting alignments provide diagnostics by showing data and resource violations and control-flow violations projected over the original log.

6 Experimental Results

We applied our approach and toolset in two case studies: one for checking compliance of a personal loan process within a global financial organization and one for analyzing the process of patient treatment in ICU department of a Dutch hospital against a medical guideline.

Case study 1: The event log related to the first case study is taken from BPI challenge of 2012[2] and it has 13.087 traces. A compliance rule restricting this process states: "loan applications with requesting amount less that 5000 and more than 50000 must not be approved". This compliance rule is of second type of data-aware compliance rules. We found 117 number of deviations from the compliance rule out of all 13.087 traces executed, i.e., 117 approved application requested an amount for more than 50000 or less than 5000.

Case study 2: In this case study we investigated the compliance of an event log taken from ICU department of a Dutch hospital with a medical guideline restricting tube feeding nutrition of patients. The event log has 1207 traces; each trace recorded the treatment that a patient received in ICU department. The compliance requirement states: " If gastric tube feeding cannot be increased then use *Domperidone* or *Metoclopramide*. The starting dosage is usually $(12 \cdot 50ml)$ and it is recommended that the increase follows the pattern $(12 \cdot 50ml, 12 \cdot 100ml, 12 \cdot 120ml)$".

The guideline is not precise here, hence we check two different data-aware rules: *i)* The nutrition must increase, else *Domperidone* or *Metoclopramide* must be administered to the patient. *ii)* the increase must follow the pattern $(12 \cdot 50ml, 12 \cdot 100ml, 12 \cdot 120ml)$.

We first checked if the nutrition has been increased for respective patients and if not, did they receive either *Domperidone* or *Metoclopramide*. We observed that from 1207 patients treated in ICU, 209 received tube feeding nutrition. In addition we found that *Metoclopramide* has not been administered for these patient; only one patient has received this medicine and it has been independent from tube feeding nutrition. For 72 patients, the tube feeding nutrition has increased without any problem. 56 patients received *Domperidone* when nutrition was not increased.

We observed in total 81 violating traces. We identified several patterns for these violations. In some of violating traces, we observed that although nutrition has increased, patients

[1] Available from `www.processmining.org`

[2] This event log is publicly available at:
`dx.doi:10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`

received *Domperidone*. It could be that that *Domperidone* was administered to these patients independent from nutrition and for other purposes.

Another group of violations are related to patients that did not receive *Domperidone* although nutrition was not increased for them. We checked these traces further and found that this violation occurs in two situations. One group contained *real violations* because patients never received *Domperidone* despite the nutrition was not increased. However for another group of patients we observed several iterations of tube feeding nutrition with an increase inside every iteration. For example a patient has received nutrition following two times the pattern $(12 \cdot 50ml, 12 \cdot 100ml, 12 \cdot 120ml)$. That is we see the increase in every occurrence of the pattern, yet the second occurrence of the pattern starts again from $(12 \cdot 50ml)$.

We also investigated if the recommended pattern of increase has been followed or not. We found out that only for less than 20% of patients the recommended pattern of the guideline is followed. From the remaining we identified 3 groups of patients. One group of course were the 56 patients that the nutrition was not increased for them. For the second group of patients, the nutrition was increased with the pattern $(12 \cdot 50ml, 12 \cdot 100ml, 12 \cdot 140ml)$. The third group mostly followed the pattern $(500ml, 1000ml, 1500ml, 1700ml)$. These patients received *Domperidone* once in every 24 hours, unlike the other groups that received *Domperidone* every two hours a day.

Applying the technique presented in this paper, we were able to check compliance of all the traces in the event logs rather than being limited to sample based compliance checking. The technique works on large event logs because we can focus on events relevant to a specific compliance rule and abstract from all other events. Our technique identified and located the deviations by visualizing for each trace the difference between admissible and observed behavior in the alignment. The extent of the deviation is reported in two ways: (1) The alignment visualizes observed and expected values of deviating events. (2) We compute statically information about the number of deviations per case, in total, etc. A domain specialists can use this information to assess their severity and analyze root-causes.

7 Related Work

Existing work in data and resource compliance checking mainly focuses on design time verification. In [6], authors incorporate data in specification of compliance rules. These rules are expressed using a query language which is an extended version of BPMN-Q and they are formalized by mapping into PLTL. They can visualize violations by indicating execution paths in a process model causing them.

In [17] the control-flow is modeled and object life-cycles model other perspectives. Later object state change becomes explicit in the process model and then the process model is used to generate life-cycles for each object type used in the process. The consistency between business process models and life-cycles of business objects is checked. Apart from the fact that this approach also focuses on verification of models, it is not discussed how deviation points represented and if further diagnostics are provided.

Other approaches such as [9,19,21] enforce compliance of business processes through the use of compliance patterns. These patterns include some of the data and resource-aware compliance requirements such as four-eye principle. However, specific checking technique are not discussed. Further on, the work in [10] addresses the verification of process models for data-aware compliance requirements. The authors do not apply a particular modeling language for specifying compliance rules, but introduce a general notion for data-aware conditions. For checking data-aware requirements, abstraction strategies are introduced to reduce complexity of data-aware compliance checking and deal with state explosion

issues. This is achieved by abstracting from concrete state of data objects. The approach automatically derives an abstract process model and an abstract compliance rule.

Process mining techniques [1] offer a means to more rigorously check compliance and ascertain the validity of information about an organization's core processes. The challenge is to compare the prescribed behavior (e.g., a process model or set of rules) to observed behavior (e.g., audit trails, workflow logs, transaction logs, message logs, and databases). Various techniques have been proposed for checking control-flow compliance rules based on event data including LTL-based checking [3] . In [13] both LTL-based and SCIFF-based (i.e., abductive logic programming) approaches are used to check compliance with respect to a declarative process model and an event log. Dozens of approaches have been proposed to check conformance given a Petri-net and an event log [2,16].

The classical data-aware conformance checking [11] that we have applied in our approach, allows for aligning event logs and process models for multi-perspective conformance checking but it has some limitations; as we can only check the compliance rules of the first category we discussed earlier. In addition if a deviation is observed for an activity that is restricted with several data attributes at the same time, the classical data-aware conformance checking can only indicate the deviation but not the specific data attribute(s) causing the deviation; resulting in less precise diagnostics. Moreover using classical data-aware conformance checking, we cannot focus only on specific rules and abstracting from other activities that are not restricted by respective compliance rule. That is we need to provide a data-aware Petri net that captures the behavior of the whole process; consequently make the checking less flexible. We have covered above mentioned limitations in our approach by extending the classical data-aware conformance checking.

8 Conclusion and Future Work

In this paper, we provided an approach for data and resource-aware compliance checking of behavior recorded in execution logs. We show a collection of different compliance constraints restricting process data and resources. In addition, we provide two generic techniques for checking these rules based on alignments. Our technique separates control-flow, data and resource compliance checking to the possible extent, and provides integrated diagnostic information about both control-flow violations, and data and resource related compliance violations. In particular, our technique is capable of showing diagnostic information about violations of a compliance rule in a process instance. We have shown this technique to be feasible for compliance rules on data dependencies between two or three data attributes. More complex rules are possible but require additional pre-processing and data-aware alignment steps. A more scalable technique is subject to further research.

We provide an implementation of our techniques in the *ComplianceFarmework* package of ProM. The software has been tested in synthetic logs and two case studies involving real-life logs from a financial institute and ICU department of a hospital. The results are encouraging: we were able to uncover various violations and no performance issues were encountered. Future research aims at making the approach more user-friendly. While the compliance checking itself is fully automatic, the user still has to create formal compliance rules and interpret the results. Formal data-aware compliance rules could be elicited in questionnaire-based approach similar to [15]. Further, a higher-level compliance language and a dashboard for integrating results of control-flow, temporal, and data and resource-aware compliance checking to provide a multi-perspective view on data are required.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
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 Mining and Knowledge Discovery 2(2), 182–192 (2012)
3. van der Aalst, W.M.P., de Beer, H., van Dongen, B.F.: Process Mining and Verification of Properties: An Approach based on Temporal Logic. In: On the Move to Meaningful Internet Systems 2005: CoopIS 2005. LNCS, vol. 3760, pp. 130–147. Springer-Verlag, Berlin (2005)
4. Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: the conformance checking case. In: SAC. pp. 1709–1716. ACM (2012)
5. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: EDOC. pp. 55–64. IEEE Computer Society (2011)
6. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: ICSOC/SW. LNCS, vol. 5900, pp. 500–515 (2009)
7. Botha, R.A., Eloff, J.H.P.: Separation of duties for access control enforcement in workflow environments. IBM Systems Journal 40(3), 666–682 (2001)
8. Elgammal, A., Türetken, O., van den Heuvel, W.J., Papazoglou, M.P.: On the formal specification of regulatory compliance: A comparative analysis. In: ICSOC Workshops. LNCS, vol. 6568, pp. 27–38 (2010)
9. Elgammal, A., Türetken, O., van den Heuvel, W.J., Papazoglou, M.P.: Root-cause analysis of design-time compliance violations on the basis of property patterns. In: ICSOC. LNCS, vol. 6470, pp. 17–31 (2010)
10. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: ER. LNCS, vol. 6412, pp. 332–346. Springer (2010)
11. de Leoni, M., van der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: BPM. LNCS, vol. 8094, pp. 113–129. Springer (2013)
12. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning event logs and declarative process models for conformance checking. In: BPM. pp. 82–97 (2012)
13. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. TWEB 4(1) (2010)
14. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did i misbehave? diagnostic information in compliance checking. In: BPM. pp. 262–278 (2012)
15. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Supporting domain experts to select and configure precise compliance rules. In: BPM Workshops. Lecture Notes in Business Information Processing, vol. 171, pp. 498–512. Springer (2013)
16. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
17. Ryndina, K., Küster, J.M., Gall, H.: Consistency of business process models and object life cycles. In: MoDELS Workshops. LNCS, vol. 4364, pp. 80–90. Springer (2006)
18. Samarati, P., di Vimercati, S.D.C.: Access control: Policies, models, and mechanisms. In: FOSAD. LNCS, vol. 2171, pp. 137–196. Springer (2000)
19. Schumm, D., Türetken, O., Kokash, N., Elgammal, A., Leymann, F., van den Heuvel, W.J.: Business process compliance through reusable units of compliant processes. In: ICWE Workshops. LNCS, vol. 6385, pp. 325–337. Springer (2010)
20. Taghiabadi, E.R., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: CAiSE. LNCS, vol. 7908, pp. 304–320. Springer (2013)
21. 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)
22. di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Access control: principles and solutions. Softw., Pract. Exper. 33(5), 397–421 (2003)