Visually Monitoring Multiple Perspectives of Business Process Compliance*

David Knuplesch[1], Manfred Reichert[1], and Akhil Kumar[2]

[1] Institute of Databases and Information Systems, Ulm University, Germany
{david.knuplesch,manfred.reichert}@uni-ulm.de
[2] Smeal College of Business, Pennsylvania State University, PA, USA
akhilkumar@psu.edu

Abstract. A challenge for any enterprise is to ensure conformance of its business processes with imposed compliance rules. The latter may constrain multiple perspectives of a business process, including control flow, data, time, resources, and interactions with business partners. However, business process compliance cannot completely be decided at design time, but needs to be monitored during run time as well. This paper introduces a comprehensive framework for visually monitoring business process compliance. As opposed to existing approaches, the framework supports the visual monitoring of all relevant process perspectives based on the extended Compliance Rule Graph (eCRG) language. Furthermore, it not only allows detecting compliance violations, but visually highlights their causes as well. Finally, the framework assists users in monitoring business process compliance and ensuring a compliant continuation of their running business processes.

Keywords: business process compliance, compliance monitoring

1 Introduction

Correctness issues of business process models have been intensively discussed for more than a decade. While early work focused on syntactical correctness and soundness constraints (e.g., absence of deadlocks and lifelocks), the compliance of business processes with semantic constraints has been increasingly considered during the recent years. Usually, *compliance rules* stem from domain-specific requirements, e.g., corporate standards or legal regulations [1], and need to be ensured in all phases of the process life cycle [2, 3]. In this context, approaches dealing with the compliance of business processes during their execution are covered by the notion of *compliance monitoring.* In general, events of running business processes need to be considered to detect run-time violations of compliance rules and to notify users accordingly (cf. Fig. 1).

In general, two kinds of compliance monitoring need to be distinguished–reactive and proactive. Regarding *reactive monitoring*, the system only reports

* This work was done within the research project C^{3}Pro funded by the German Research Foundation (DFG) under project number RE 1402/2-1.

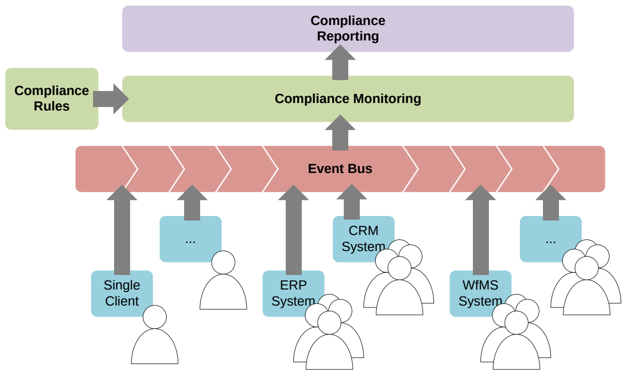
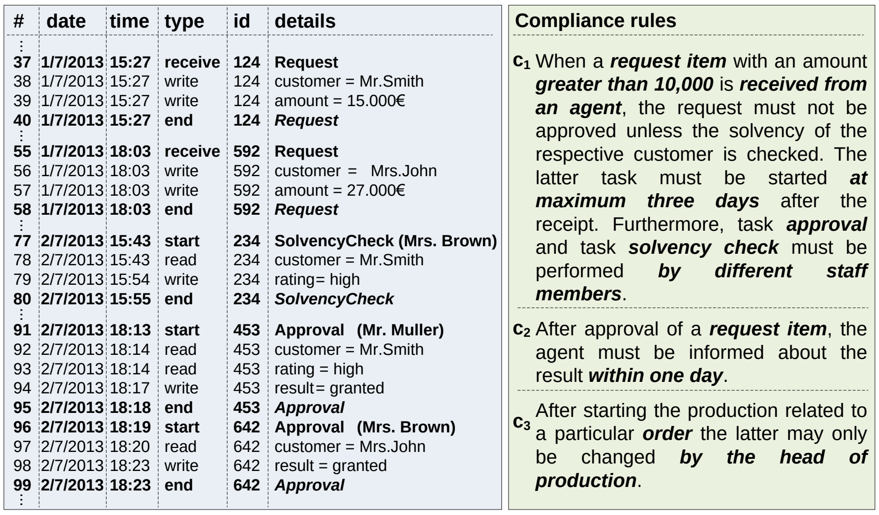


Fig. 1: Compliance Monitoring [4]

a compliance violation once it has occurred. By contrast, *proactive monitoring* aims to preventively avoid any violation; e.g., by recommending appropriate tasks, which still need to be executed to meet the compliance rule, to users.

As example consider the event log from Fig. 2, which refers to an order-to-delivery process [5]: Compliance rule c_1, shown on the right, is satisfied in one case, but violated in another. In particular, the depicted log refers to two different request items related to customers *Mr. Smith* and *Mrs. John*. These items, in turn, trigger two different instances of compliance rule c_1. In both cases, the amount is greater than 10,000 € and hence a solvency check is required (cf. c_1). However, the latter was only performed for the request item of Mr. Smith, but not for the one of *Mrs. John*, i.e., c_1 is violated in the latter case. In addition to the violation of c_1, compliance rule c_2 is violated twice. While the violated



#	date	time	type	id	details
⋮					
37	**1/7/2013**	**15:27**	**receive**	**124**	**Request**
38	1/7/2013	15:27	write	124	customer = Mr.Smith
39	1/7/2013	15:27	write	124	amount = 15.000€
40	**1/7/2013**	**15:27**	**end**	**124**	***Request***
⋮					
55	**1/7/2013**	**18:03**	**receive**	**592**	**Request**
56	1/7/2013	18:03	write	592	customer = Mrs.John
57	1/7/2013	18:03	write	592	amount = 27.000€
58	**1/7/2013**	**18:03**	**end**	**592**	***Request***
⋮					
77	**2/7/2013**	**15:43**	**start**	**234**	**SolvencyCheck (Mrs. Brown)**
78	2/7/2013	15:43	read	234	customer = Mr.Smith
79	2/7/2013	15:54	write	234	rating= high
80	**2/7/2013**	**15:55**	**end**	**234**	***SolvencyCheck***
⋮					
91	**2/7/2013**	**18:13**	**start**	**453**	**Approval (Mr. Muller)**
92	2/7/2013	18:14	read	453	customer = Mr.Smith
93	2/7/2013	18:14	read	453	rating = high
94	2/7/2013	18:17	write	453	result= granted
95	**2/7/2013**	**18:18**	**end**	**453**	***Approval***
96	**2/7/2013**	**18:19**	**start**	**642**	**Approval (Mrs. Brown)**
97	2/7/2013	18:20	read	642	customer = Mrs.John
98	2/7/2013	18:23	write	642	result = granted
99	**2/7/2013**	**18:23**	**end**	**642**	***Approval***
⋮					

Compliance rules

c_1 When a ***request item*** with an amount ***greater than 10,000*** is ***received from an agent***, the request must not be approved unless the solvency of the respective customer is checked. The latter task must be started ***at maximum three days*** after the receipt. Furthermore, task ***approval*** and task ***solvency check*** must be performed ***by different staff members***.

c_2 After approval of a ***request item***, the agent must be informed about the result ***within one day***.

c_3 After starting the production related to a particular ***order*** the latter may only be changed ***by the head of production***.

Fig. 2: Event log of order-to-delivery processes and compliance rules

instance of rule c_1 will never be successfully completed, the violations of c_2 still may be healed by informing the *agent* about the results of the approvals. The compliance rule examples further indicate that solely monitoring control flow dependencies between tasks is not sufficient to ensure compliance at run time. In addition, constraints with respect to the data, time, and resource perspectives of a business process as well as the interactions this process has with partner processes need to be monitored as well [6–9]. For example, the data perspective of compliance rule c_1 is addressed by the *request item* and its *amount.* In turn, receiving the *request item* (cf. c_1) corresponds to an interaction with a business partner. Furthermore, the phrase "*by different staff members*" deals with the resource perspective, whereas the condition "*at maximum three days*" refers to the time perspective. To meet practical demands, compliance monitoring must not omit these process perspectives.
Altogether, the following requirements need to be addressed:

RQ1. As a fundamental challenge of any compliance monitoring approach, compliance violations must be reliably detected and reported to the appropriate parties by using alerts, emails, text messages, or other notification mechanisms. Furthermore, compliance-aware user guidance is needed to avoid rule violations.

RQ2. Since compliance is not restricted to the control flow perspective solely, the time, resource and data perspectives of a business process as well as its interactions with business partners need to be considered during compliance monitoring as well.

RQ3. In general, the execution of a business process may trigger multiple instances of the same compliance rule. On one hand, this highlights the need for being able to identify the causes of a specific compliance violation as well as for providing proper user feedback [10]. On the other, this mightlead to situations in which a compliance rule is fulfilled or violated multiple times in the context of a particular process instance. Accordingly, any compliance assessment must reflect the relation between fulfilled and violated instances of compliance rules.

RQ1-RQ3 cover the essential *compliance monitoring functionalities* (CMFs) as proposed in [4]. Therefore, they may be used to compare existing approaches for monitoring business compliance. However, [4] also states that existing approaches only partially meet the CMFs. In particular, the combination of an expressive language (RQ2) and full traceability (RQ3) is not well understood yet.

This paper extends the work, we presented in [5] in order to provide a comprehensive framework addressing RQ1-RQ3. In particular, it adds detailed algorithms for compliance rule monitoring based on the visual *extended Compliance Rule Graph (eCRG)* language [8, 9]. The current state of a particular eCRG is reflected through a set of visual rule markings. The latter not only indicate compliance violations, but may also be utilized for recommending the next process tasks to be executed to ensure compliance (RQ1). Furthermore, these markings allow us to clearly differ between fulfilled and violated instances of an eCRG and also provide a suitable basis for compliance metrics (RQ3). Note that the eCRG language supports the time, resource and data perspectives as well as interactions with business partners (RQ2). We evaluate the algorithms based on

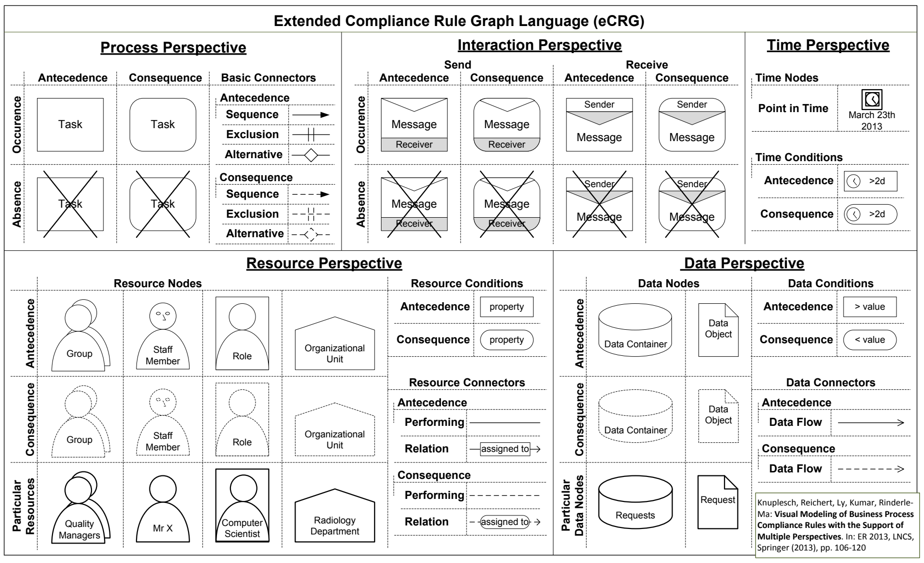


Fig. 3: Elements of the eCRG language [8, 9]

a proof-of-concept prototype, which was also applied to real world compliance scenarios we had obtained from one of our case studies in the healthcare domain [9].

The remainder of this paper is organized as follows: Section 2 introduces the *extended Compliance Rule Graph* (eCRG) language. The monitoring framework as well as algorithms that manage the markings of an eCRG are introduced in Section 3. Section 4 validates the framework and presents its proof-of-concept prototype. Section 5 discusses related work. Section 6 concludes the paper and provides an outlook on future research.

2 Fundamentals

This paper utilizes the extended Compliance Rule Graph (eCRG) language we developed for modeling compliance rules [8, 9]. The eCRG language is based on the Compliance Rule Graph (CRG) language [10]. As opposed to CRG, eCRG not only focuses on the control flow perspective, but also provides integrated support for the resource, data and time perspectives as well as for the interactions with business partners. To cover the various perspectives, the eCRG language allows for *attachments* in addition to *nodes* and *connectors* (i.e. edges). Nodes, connectors and attachments may be partitioned into an *antecedence pattern* and one or several related *consequence patterns*. Both patterns are modeled using *occurrence* and *absence nodes*, which either express the occurrence or absence of certain events (e.g. related to the execution of a particular task) or which refer to process entities (e.g. data objects). In turn, edges and attachments are used to

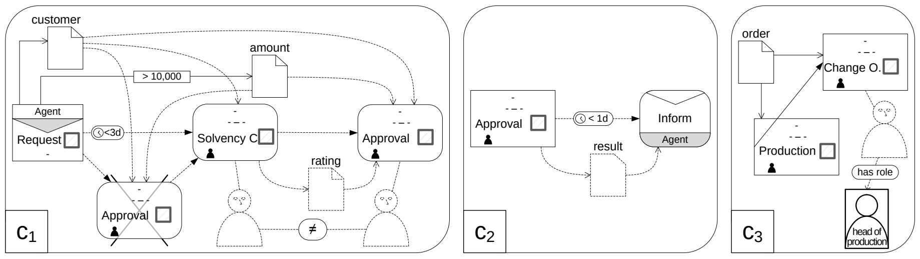


Fig. 4: Modeling compliance rules $c_1 - c_3$ with the eCRG language

refine the specification of the elements they are affiliated to (e.g., by specifying control flow dependencies). Furthermore, an eCRG may contain *instance nodes* referring to particular objects that exist independently from the respective rule (e.g. *Mr. Smith*, *postnatal ward*, *physician*). Note that instance nodes are neither part of the antecedence nor the consequence pattern. Fig. 3 provides an overview of eCRG elements, which are applied in Fig. 4 to model the compliance rules from Fig. 2. In this paper, we refer to the following elements of an eCRG:

- *Nodes*: These include, for example, *TaskNodes*, *MessageNodes*, *PointInTimeNodes*, *DataObjects*, and *ResourceNodes*.
- *Edges*: These include, for example, *SequenceFlowEdges*, *DataFlowEdges*, *PerformingEdges*, *ResourceRelations*, and *DataRelations*.
- *Attachments*: These include, for example, *DataConditionAttachments*, *ResourceConditionAttachments*, and *TimeConditionAttachments*.

In this context, two elements a and b of an eCRG have the *same dependency level* ($a \triangleq b$), if they are elements of the same pattern. In turn, an attachment or edge c *corresponds to* a node d, if c directly or indirectly constrains d. Finally, set $\Lambda := Nodes \cup Edges \cup Attachments$ contains all elements of an eCRG. For a more formal eCRG specification, we refer to [11, 12].

3 eCRG Compliance Monitoring

This section introduces the framework for visually monitoring multiple perspectives of business process compliance at runtime. As discussed in Sect. 1, compliance monitoring is based on streams of events occurring during the execution of business processes. In particular, it aims to determine or prevent compliance violations. For this purpose, the framework annotates and marks the elements of an eCRG with text, colors and symbols during the processing of events. These markings not only provide a basis for determining the state of compliance of a particular rule, but also highlight the causes of occurring compliance rule violations.

States of Compliance Rules. When monitoring the compliance of running processes, compliance rule instances may be in different states. Fig. 5 outlines the

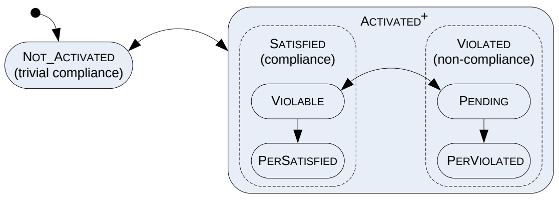


Fig. 5: States of compliance rules

states supported by the framework. The most fundamental state is NOT_ACTIVATED, i.e., the compliance rule does not apply to the running process instance so far. In turn, state ACTIVATED expresses that the compliance rule is applicable to the process instance. Furthermore, this state includes the sub-states SATISFIED and VIOLATED (cf. Fig. 5). State SATISFIED is further partitioned into sub-states VIOLABLE and PERSATISFIED (i.e., permanently satisfied), whereas state VIOLATED includes sub-states PENDING and PERVIOLATED (i.e., permanently violated). As explained in the context of the example, business processes may trigger (i.e. activate) multiple instances of a compliance rule. Hence, a compliance rule may be in state ACTIVATED multiple times as indicated by superscript "+" in Fig. 5. However, each of these *activations* of a compliance rule may be in a different sub-state. For example, the event log of the example from Fig. 2 activates compliance rule c_1 twice (cf. Fig. 2). While the first activation is in state PERSATISFIED, the second one is in final state PERVIOLATED.

Events. As the framework enables compliance monitoring for multiple process perspectives (cf. RQ2), it not only monitors events referring to the start and end of tasks. In addition, it considers events that correspond to the sending and receiving of messages as well as data flow events. Furthermore, events may include temporal information as well as information about involved resources. Table 1 summarizes the event types supported by the framework. Each entry refers to the time the event occurred and to a unique identifier. The latter enables us to correlate start, end and data flow events of the same task or message. Note that we presume correct event streams; i.e., they do not deviate from the real process. Further, events are provided in ascending order.

eCRG Markings. To monitor the state of a compliance rule, we mark the elements of an eCRG (cf. Sect. 2, [8, 9]) with symbols, colors and text (cf. Fig. 6). Such a *marking of an eCRG* highlights whether or not the events corresponding to a node have occurred so far. Further, a marking describes whether conditions corresponding to edges and attachments are satisfied, violated, or still may be evaluated.

Table 1: Supported Events

Task events	**Message events**	**Data flow events**
`start`(time, id, tasktype, performer)	`send`(time, id, message)	`write`(time, id, value$\xrightarrow{\text{param}}$source)
`end`(time, id, tasktype, performer)	`receive`(time, id, message)	`read`(time, id, value$\xleftarrow{\text{param}}$source)
	`end`(time, id, message)	

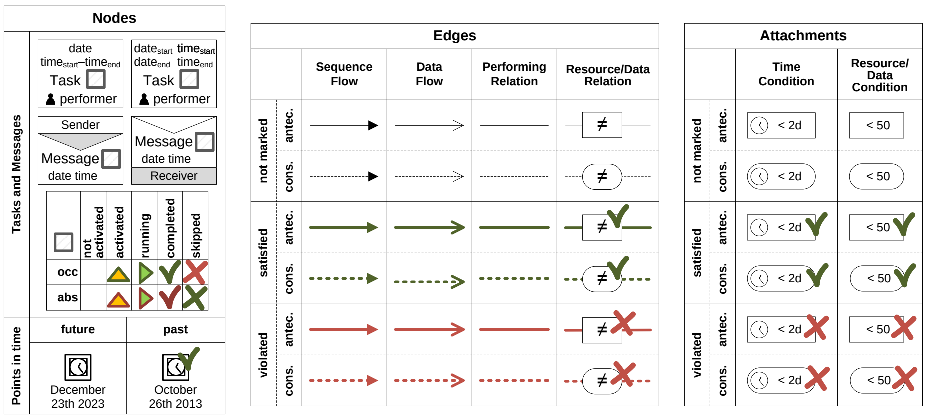


Fig. 6: Fundamental markings of eCRG elements

Let $\mathcal{R}$ be the set of resources, Ω be the set of data objects, $\mathcal{I}$ be the set of identifiers, and $\mathfrak{T}$ be the set of point in times. Further, let ϵ be the *empty value*. Then: A marking M can be described with the following functions:

– $M.mark : \Lambda \rightarrow \{\square = \epsilon, \triangle, \blacktriangleright, \checkmark, \times\}$ marks the elements of the eCRG as *not-marked* $\square$, *activated* $\triangle$, *running* $\blacktriangleright$, *satisfied* (or *completed*) $\checkmark$, and *violated* (or *skipped*) $\times$ (cf. Fig. 6),
– $M.res : \Lambda \rightarrow \mathcal{R} \cup \{\epsilon\}$ assigns resources to the elements of the eCRG,
– $M.val : \Lambda \rightarrow \Omega \cup \{\epsilon\}$ assigns values to each element of the eCRG,
– $M.id : \Lambda \rightarrow \mathcal{I} \cup \{\epsilon\}$ assigns unique identifiers to the elements of the eCRG,
– $M.start(M.end) : \Lambda \rightarrow \mathfrak{T} \cup \{\epsilon\}$ assigns starting (ending) times to the elements.

The functions of the *initial marking* 0 assign ϵ (and $\square$ respectively) to all elements of an eCRG, except the ones of the *instance pattern* that are mapped to the particular resource, data value or point in time they refer to. Since there may be multiple activations of a particular compliance rule, the *state of an eCRG* is a set $\mathcal{M}$ of markings.

Fig. 7 shows two markings for compliance rule c_1 from Fig. 2. On the left, marking F highlights the fulfillment of c_1 for the request of Mr. Smith. In turn, marking K on the right emphasizes how markings support users in proactively ensuring compliance. In particular, K indicates which data values the task *solvency check* shall read and how task *approval* shall be performed afterwards in order to satisfy c_1.

Event Processing. This section describes the processing of events with an eCRG.[3] Fig. 8 provides an overview. First, all markings are prepared for the processing. Second, effects of these preparations (i.e., changed markings) are propagated onto connected elements. Third, the actual *event handling* takes place. Fourth,

[3] [11] provides a formal specification of the operational semantic of the eCRG language.

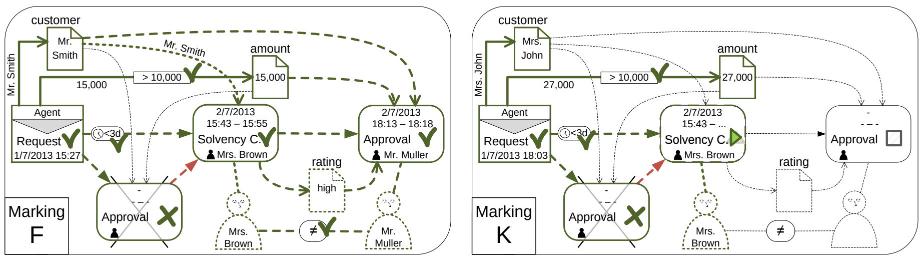


Fig. 7: Examples of markings for compliance rule c_1

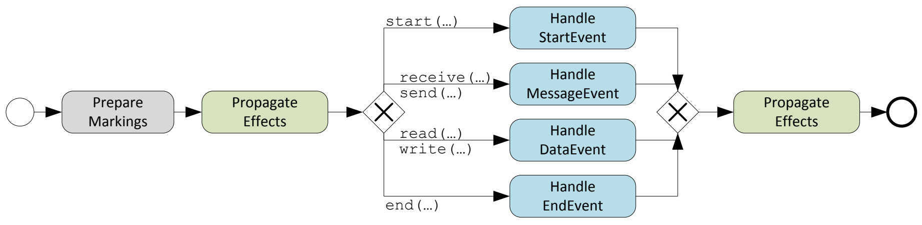


Fig. 8: Processing of start, message, data, and end events

effects of the latter step are propagated to connected elements as well. Note that the first two steps may be applied without the last two ones, e.g., to calculate the current state of a compliance rule at an arbitrary point in time.

In general, not only the occurrence of events, but also elapsing time can violate compliance, e.g, when the maximum time distance between two tasks becomes violated. To ensure that related issues are not ignored, Listing 1 updates the time perspective of markings before the latter process an event. In particular, point in time nodes are changed to ✓, if they lie in the past now, whereas time condition attachments on task nodes or sequence flow edges are skipped (✕) if they are no longer satisfiable.

Listing 2 deals with the handling of start and message events. In particular, markings of activated task or message nodes, which match the event, are re-set from △ to ▶. Accordingly, identifiers, resources and starting times are set. Note

Listing 1: Prepare Markings (with respect to the time perspective)

```
prepareMarking($M$, event(time,...))
    ForEach($pitn \in PointInTimeNodes$ with $M.mark(pitn) = \square$)
        If ($pitn \leq$ time)  $M.mark(pitn) := \checkmark$;
    ForEach($tc \in TimeConditionAttachments$ with $M.mark(tc) = \square$)
        If ($tc$ is attached to $tn \in TaskNodes$ and $M.mark(tn) = \blacktriangleright$)
            If ($\forall t \geq$ time: $tc(t_s(t), t) = \texttt{false}$)  $M.mark(tc) := \times$;
        ElseIf ($tc$ is attached to $sf = (n_1, n2) \in SequenceFlowEdges$ and $M.mark(n_1) = \checkmark$)
            If ($\forall t \geq$ time: $tc(t_e(n_1), t) = \texttt{false}$)  $M.mark(tc) := \times$;
    Return $M$;
```

Listing 2: Handle Events

```
handleStartEvent($M$, start/send/receive(time, id, type, performer))
    $\mathcal{M} := \varnothing$;
    ForEach($\sigma \subseteq \{tn | tn \in TaskNodes$ and $M.mark(tn) = \triangle$ and $typeOf(tn) =$type$\}$ )
        ForEach($tn \in \sigma$)
            $M' := copy(M)$;
            $M'.mark(tn) := \blacktriangleright$; $M'.start(tn) :=$time;
            $M'.id(tn) =$id; $M'.res(tn) :=$performer;
        $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
    Return $\mathcal{M}$;
handleMessageEvent($M$, send/receive(time, id, type))
    $\mathcal{M} := \varnothing$;
    ForEach($\sigma \subseteq \{tn | tn \in MessageNodes$ and $M.mark(tn) = \triangle$ and $typeOf(tn) =$type$\}$ )
        ForEach($tn \in \sigma$)
            $M' := copy(M)$;
            $M'.mark(tn) := \blacktriangleright$; $M'.start(tn) :=$time; $M'.id(tn) =$id; ;
        $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
    Return $\mathcal{M}$;
handleEndEvent($M$, end(time, id, type, performer))
    ForEach($tn \in TaskNodes$ with $M.id(tn) =$ id)
        $M.mark(tn) := \checkmark$; $M.end(tn) :=$time;
    Return $\{M\}$;
handleDataEvent($M$, write/read(time, id, value $\xrightarrow{\text{param}}$ source))
    ForEach($df = (n, x) \in DataflowEdges$ with $M.id(n) =$ id and $M.mark(n) = \blacktriangleright$)
        If ($typeOf(df) =$param)
            $M.mark(df) := \checkmark$; $M.val(df) :=$value;
    Return $\{M\}$;
```

that start and message events are handled non-deterministically; i.e., the changes are applied to copies of the original marking that is maintained (cf. Fig. 9.2). Further, Listing 2 specifies the handling of data events. In particular, the corresponding data flow edges of running task (message) nodes are annotated with the data value passed (cf. Figs. 9.4 and 9.5). Finally, the handling of end events is addressed in Listing 2 as well. In particular, the markings of corresponding nodes in state running (▶) are set to completed (✓); their ending time is set accordingly. (cf. Fig. 9.A)

Effects of preparing and handling events must be propagated to ensure correct markings (e.g., activation of subsequent task nodes) as well as to detect contradictory markings related to the data and resource perspectives. In particular, data values are propagated along data flow edges to connected data objects. In turn, resources are propagated from task nodes via resource edges to connected resource nodes. The propagation fails, if a resource or data object node is set to a different value before. In this case, the respective edge is skipped (✕). Furthermore, conditions and relations are evaluated as soon as possible. If any element of the eCRG, which corresponds to a task or message node, becomes skipped (e.g., due to a failed data or resource propagation, or a violated condition), the task or message node will be skipped as well. Then, outgoing sequence flows of

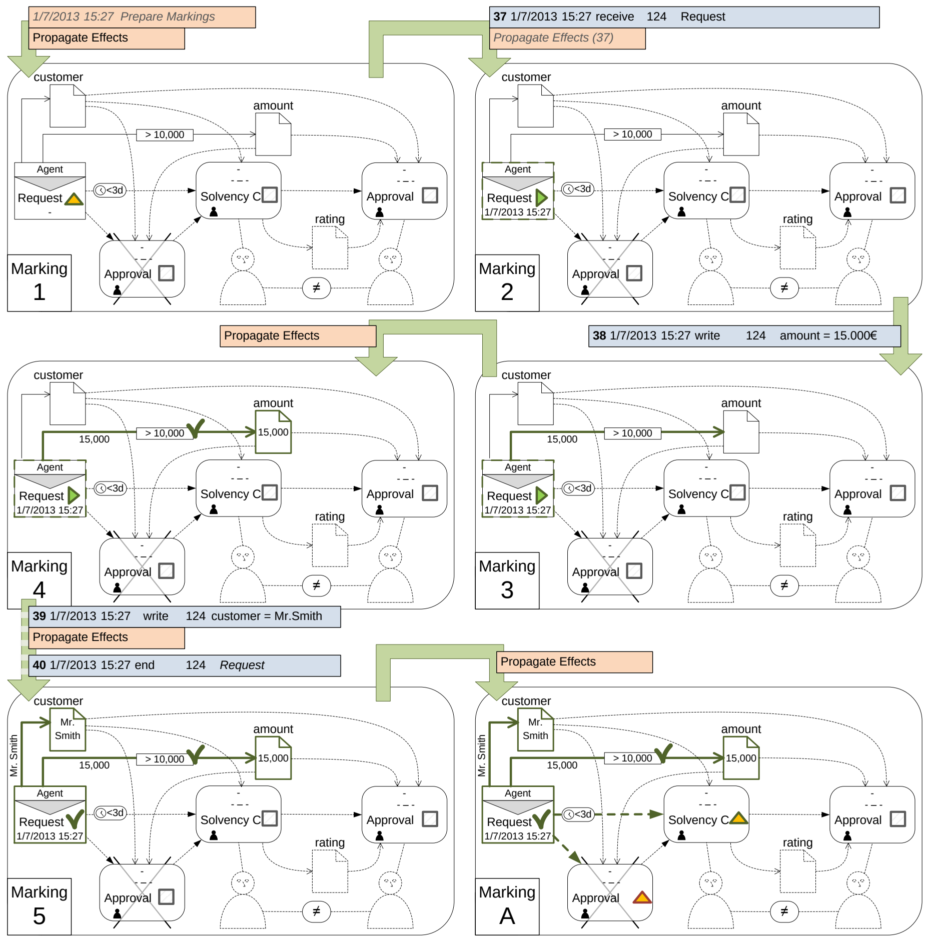

1/7/2013 15:27 Prepare Markings
Propagate Effects
37 1/7/2013 15:27 receive 124 Request
Propagate Effects (37)
Propagate Effects
38 1/7/2013 15:27 write 124 amount = 15.000€
39 1/7/2013 15:27 write 124 customer = Mr.Smith
Propagate Effects
40 1/7/2013 15:27 end 124 Request
Propagate Effects

Fig. 9: Handling of events

completed nodes will be marked as satisfied (✓). In turn, non-marked incoming edges of already started nodes as well as edges from and to skipped nodes will be skipped. Task and message nodes will be activated (△) when all incoming sequence flows, these nodes depend on, are satisfied. In turn, task or message nodes will be skipped (✕) if they depend on sequence flows being skipped as well. Note that the latter might result in the cascading skipping of other sequence flow edges (cf. Listing 3).

Table 2 illustrates the set of markings that results after processing the event stream from Fig. 2 for compliance rule c_1. In turn, Figs. 10-13 highlight conflicts regarding the data (Fig. 10), control flow (Fig. 12), resource (Fig. 11), and time (Fig. 13) perspectives. Note that conflicting markings only highlight why the considered events do not constitute a fulfillment of a particular compliance rule, but they do not necessarily lead to a violation of the latter.

Listing 3: Propagate Effects

```
effectPropagation($M$)
    ForEach($pfr = (tn, r) \in PerformingEdges$ with $M.mark(pfr) = \square$ and $M.res(tn) \neq \epsilon$)
        $M.mark(pfr) := \checkmark$; $M.res(pfr) := M.res(tn)$;
        If($M.mark(r) = \square$)
            If($r \hat{=} pfr$) $M.mark(r) := \checkmark$; $M.res(r) := M.res(tn)$;
        ElseIf ($M.res(pfr) \neq M.res(r)$) $M.mark(pfr) := \times$;
    ForEach($rr = (r_1, r_2) \in ResRelations$ with $M.mark(r_1) = M.mark(r_2) = \checkmark$)
        If($rr(r_1, r_2) = \texttt{true}$) $M.mark(rr) = \checkmark$ Else $M.mark(rr) = \times$
    ForEach($df = (n, o) \in DataFlowEdges$ with $M.mark(df) = \checkmark$)
        If($M.mark(o) = \square$)
            If($o \hat{=} df$) $M.mark(o) := \checkmark$; $M.val(o) := M.val(df)$;
        ElseIf ($M.val(df) \neq M.val(o)$) $M.mark(df) := \times$;
    ForEach($dr = (o_1, o_2) \in DataRelations$ with $M.mark(o_1) = M.mark(o_2) = \checkmark$)
        If($dr(o_1, o_2) = \texttt{true}$) $M.mark(dr) := \checkmark$ Else $M.mark(dr) := \times$
    ForEach($att \in Attachments$ with $M.mark(att) = \square$ and $M.mark(@(att)) = \checkmark$)
        If($att(@(att)) = \texttt{true}$) $M.mark(att) := \checkmark$ Else $M.mark(att) := \times$;
    ForEach($x, y \in AllElements$ with $x \hat{=} y$ and $y$ corresponds to $x$)
        If($M.mark(y) = \times$ and $M.mark(x) \neq \times$) $M.mark(x) := \times$;
    ForEach($sf = (n_1, n2) \in SequenceFlowEdges$ with $M.mark(sf) = \square$)
        If($M.mark(n_1) = \checkmark$) $M.mark(sf) = \checkmark$;
        If($M.mark(n_1) = \times$ or $M.mark(n_2) \in \{\blacktriangleright, \checkmark, \times\}$) $M.mark(sf) = \times$;
    ForEach($n \in TaskNodes \cup MessageNodes$ with $M.mark(n) = \square$)
        If($\forall sf = (n, n_2) \in SequenceFlowEdges$ with $sf \hat{=} n$ holds $M.mark(sf) = \checkmark$)
            $M.mark(n) = \triangle$;
    Repeat
        $M' = M$;
        ForEach($sf = (n_1, n2) \in SequenceFlowEdges$ with $M.mark(sf) = \square$)
            If($M.mark(n_1) = \times$ or $M.mark(n_2) \in \{\blacktriangleright, \checkmark, \times\}$) $M.mark(sf) = \times$;
        ForEach($n \in TaskNodes \cup MessageNodes$ with $M.mark(n) = \square$)
            If($\exists sf = (n, n_2) \in SequenceFlowEdges$ with $sf \hat{=} n$ and $M.mark(sf) = \times$)
                $M.mark(n) = \times$;
            If($\exists sf = (n_2, n) \in SequenceFlowEdges$ with $sf \hat{=} n$ and $M.mark(sf) = \times$)
                $M.mark(n) = \times$;
    Until ($M = M'$);
    Return $M$;
```

Compliance assessments and metrics. Based on the described set of markings, we can identify the different *activations* of an eCRG and derive their state of compliance. In turn, *activations* correspond to the minimal markings, which satisfy the antecedence pattern, but do not satisfy any element of the antecedence absence pattern (cf. Sect. 2). In particular, an activation is satisfied if there exists another marking extending the activation and satisfying the consequence pattern. We omit a formal specification here and refer to [11] instead.

Table 3 highlights the properties of both activated markings A and B along the log from Fig. 2. In particular, Table 3 shows that c_1 is activated twice; once satisfied and once violated. Furthermore, Table 3 indicates the events that complete the activations (39 and 58), the fulfillment (95), and the violation (99).

Table 2: Markings after processing the event log from Fig. 2

#	Request		Approval (CA)			Solvency Check			Approval (CO)			cust.	cust.→ App.(CA)	cust.→ Solv.C	amount	rating
1	△		□			□			□			ϵ	ϵ	ϵ	ϵ	ϵ
A!	✓	124	△			△			□			Smith	ϵ	ϵ	15.000	ϵ
B!	✓	592	△			△			□			John	ϵ	ϵ	27.000	ϵ
C	✓	124	×			✓	234	Brown	△			Smith	ϵ	Smith	15.000	high
D	✓	592	△			×	234	Brown	×			John	ϵ	Smith	27.000	ϵ
E	✓	592	×	453	Muller	△			□			John	Smith	ϵ	27.000	ϵ
F	✓	124	×			✓	234	Brown	✓	453	Muller	Smith	ϵ	Smith	15.000	high
G	✓	124	✓	453	Muller	△			□			Smith	Smith	ϵ	15.000	high
H	✓	124	×			✓	234	Brown	×	642	Brown	Smith	ϵ	Smith	15.000	ϵ
I	✓	592	✓	642	Brown	△			□			John	John	ϵ	27.000	ϵ
J	✓	124	×	642	Brown	△			□			Smith	John	ϵ	25.000	ϵ

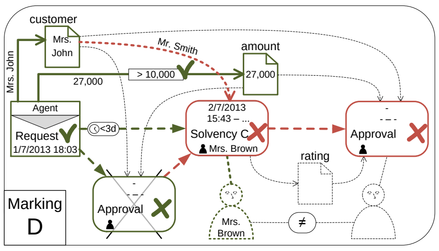


Fig. 10: Data conflict

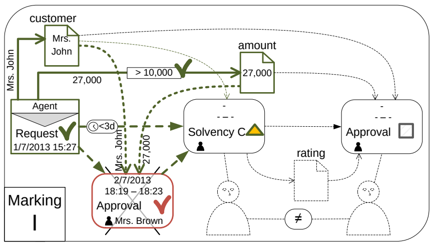


Fig. 12: Control flow conflict

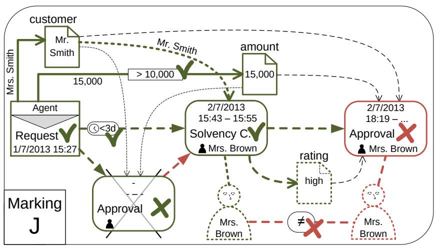


Fig. 11: Resource conflict

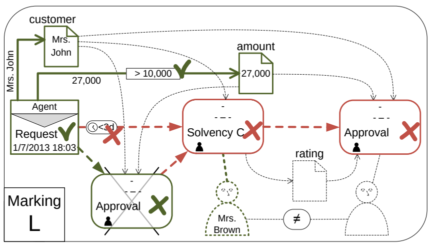


Fig. 13: Time conflict

Note that it is easy to specify metrics based on the states of compliance based on the number of activated markings in a particular compliance state PROPERTY: $\#(\mathcal{M}, \textsc{Property}) := |\{M \in \mathcal{M} | \textsc{Property}(M)\}|$; e.g., Table. 3 refers to the

$$\textit{compliance rate } \mu_1 := \frac{\#(\mathcal{M},\textsc{Satisfied})}{\#(\mathcal{M},\textsc{Activated})} \quad \text{and}$$

$$\textit{critical rate } \mu_2 := \frac{\#(\mathcal{M},\textsc{Violable})+\#(\mathcal{M},\textsc{Pending})}{\#(\mathcal{M},\textsc{Activated})}.$$

Table 3: Compliance assessments and metrics

#	Extensions	ACTIVATED	VIOLABLE	PENDING	PERSATISF.	PERVIOL.
A	{C,F,G,J}	39-...		39-95	95-...	
B	{D,E,H,I}	58-...		58-99		99-...

date	time	μ_1	μ_2
1/7/2013	15:00	n.d.	n.d.
2/7/2013	15:00	0%	100%
2/7/2013	18:18	50%	50%
2/7/2013	19:00	50%	0%

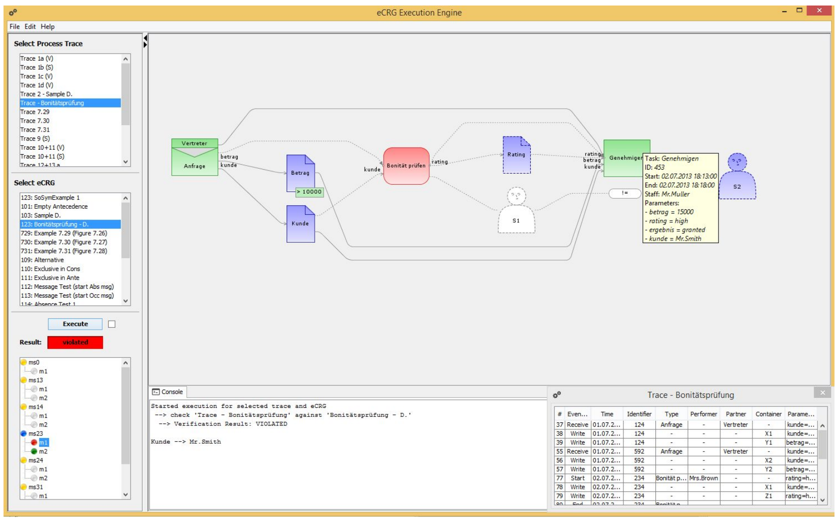
4 Evaluation

The eCRG language has been evaluated with respect to different aspects (cf. [8, 12]). In particular, its expressiveness allows modeling different sets of compliance patterns (e.g. [13]). In turn, a case study in the medical domain revealed that a business analyst was able to properly use eCRG [9]. Finally, current empirical studies indicate that there is *no significant difference* between computer experts and business analysts in understanding eCRGs.

To verify the feasibility of the presented compliance monitoring framework, we implemented an advanced proof-of-concept prototype [14]. The latter incrementally processes event logs, unfolds the markings (cf. Sect. 3), and visualizes them. Note that the prototype supports additional features, not discussed in this paper due to space limitations; e.g., beyond end-start control flow constraints, start-start, start-end, and end-end constraints are supported as well. We applied the prototype to different scenarios including the presented order-to-delivery example as well as real world compliance scenarios obtained in the context of a case study in the healthcare domain [9]. Note that the benefits of the framework come with the cost of a high, up-to exponential computational complexity of $O(|\text{EVENTS}|^{|\text{NODES}|})$. Fig. 14 provides a screenshot of the eCRG execution engine.

5 Related Work

In recent years, business process compliance has gained increasing attention and several surveys have been provided [15, 7, 16]. Accordingly, interest in *compliance monitoring* and *continous auditing* [17] has increased as well. [18] enriches process models with a semantic layer of internal controls. In [19, 20], the detailed architecture of an online auditing tool (OLAT) is described. The latter allows



Task: *Genehmigen*
ID: *453*
Start: *02.07.2013 18:13:00*
End: *02.07.2013 18:18:00*
Staff: *Mr.Muller*
Parameters:
- *betrag = 15000*
- *rating = high*
- *ergebnis = granted*
- *kunde = Mr.Smith*

```
Started execution for selected trace and eCRG
 --> check 'Trace - Bonitätsprüfung' against 'Bonitätsprüfung - D.'
  --> Verification Result: VIOLATED

Kunde --> Mr.Smith
```

Trace - Bonitätsprüfung

#	Even...	Time	Identifier	Type	Performer	Partner	Container	Parame...
37	Receive	01.07.2...	124	Anfrage	-	Vertreter	-	kunde=...
38	Write	01.07.2...	124	-	-	-	X1	kunde=...
39	Write	01.07.2...	124	-	-	-	Y1	betrag=...
55	Receive	01.07.2...	592	Anfrage	-	Vertreter	-	kunde=...
56	Write	01.07.2...	592	-	-	-	X2	kunde=...
57	Write	01.07.2...	592	-	-	-	Y2	betrag=...
77	Start	02.07.2...	234	Bonität p...	Mrs.Brown	-	-	rating=h...
78	Write	02.07.2...	234	-	-	-	X1	kunde=...
79	Write	02.07.2...	234	-	-	-	Z1	rating=h...

Fig. 14: Proof-of-concept implementation

Table 4: Compliance Monitoring Functionalities [4]

Approach	CMF 1 time	CMF 2 data	CMF 3 resources	CMF4 non atomic	CMF 5 life-cycle	CMF 6 multi-instance	CMF 7 reactive mgmt	CMF 8 proactive mgmt	CMF 9 root cause	CMF 10 compl. degree
Mubicon LTL [24]	+/-	-	-	+	-	-	+	+	+	+/-
Mubicon EC [36]	+	+/-	+	+	+	+	+	-	+/-	+/-
ECE Rules [37]	+	+/-	+	+	-	-	+	-	+/-	+
SCT [22]	+/-	-	+	+	+	-	-	+	-	-
SeaFlows [10]	+/-	+/-	+/-	+	+/-	+	+	+	+	+/-
eCRG Monitoring	+	+	+	+	+	+	+	+	+	+

monitoring the operations of an organization in detective, corrective and preventive modes. The broad spectrum of techniques enabling compliance monitoring include *behavioural profiles* [21] (i.e., to utilize ordering relations), *Supervisory Control Theory* [22] (i.e., to prevent from actions leading to compliance violations), and visual declarative constraints [23], which are transformed into *Event Calculus* and *LTL*. To enable fine-grained compliance diagnostics at run-time, *Compliance Rule Graphs* [10] and colored automata [24] are utilized, focusing on control flow. Finally, [4] compares approaches for monitoring business compliance based on 10 *compliance monitoring functionalities* (CMF). In particular, it emphasizes that none of the existing approaches provides a satisfactory solution that combines an expressive language with full traceability (cf. RQ2+RQ3). In turn, the presented approach for monitoring compliance with the eCRG language supports all 10 CMFs (cf. Table 4) [4].

However, [25, 13, 26] *a posteriori* verify the compliance of execution logs with a set of constraints. Some approaches not only focus on the control flow perspective, but consider other perspectives as well. *A priori* or *design time* compliance checking is addressed by a multitude of approaches, which commonly apply model checking; e.g., [27–30]. Some of them use visual compliance rules and address multiple perspectives. To specify compliance rules, formal languages (e.g., LTL [28]), pattern-based approaches (e.g., [31, 13]) are applied. Further, visual notations [27, 10] as well as methodologies to relate the latter with informal and textual specifications [32] have been proposed. Note that declarative languages [33, 34] can be also applied to specify compliance rules. Finally, the integration of business process compliance throughout the entire process lifecycle [2, 3] as well as monitoring of performance measures in the context of artifact-centric process models in real-time [35] have been addressed.

6 Summary and Outlook

In recent years, business process compliance has gained an increasing interest. A multitude of approaches focus on compliance monitoring at run time [18, 17, 19, 10, 24, 36]. However, existing approaches do not provide a satisfactory solution that combines an expressive language with full traceability [4].

To remedy this drawback, we proposed, developed and demonstrated a compliance monitoring framework that utilizes the extended compliance rule graph (eCRG) language, which enables the visual modeling of compliance rules with the support of the control flow, data, time, and resource perspectives as well as interactions with partners (RQ2). In particular, the presented approach marks eCRG with text, color and symbols to visually highlight the current state of compliance, whereas the informally presented operational semantics specifies how observed events evolve these markings (RQ1) Finally, formal criteria for compliance assessments are provided in a related report [11] and compliance metrics were introduced (RQ3). As opposed to existing approaches, the framework combines full traceability with an expressive visual notation. Moreover, we provide a proof-of-concept implementation that was applied to different scenarios.

Beyond the identification and highlighting of particular compliance violations in detail, another important task is to summarize and present the latter in abstract compliance reports. Hence, we aim at a user-friendly navigation through different levels of granularity. Furthermore, we will conduct further empirical studies as well as usability experiments.

References

1. Governatori, G., Sadiq, S.: The journey to business process compliance. In: Handbook of Research on BPM. IGI Global (2009) 426–454
2. Knuplesch, D., Reichert, M.: Ensuring business process compliance along the process life cycle. Technical Report 2011-06, Ulm University (2011)
3. Ly, L.T., Rinderle, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. Data & Knowledge Engineering **64**(1) (2008) 3–23
4. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: A framework for the systematic comparison and evaluation of compliance monitoring approaches. In: EDOC'13, IEEE (2013) 7–16
5. Knuplesch, D., Reichert, M., Kumar, A.: Towards visually monitoring multiple perspectives of business process compliance. In: CAiSE'15 Forum, CEUR-WS 41–48
6. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Hints on how to face business process compliance. In: JISBD'10. (2010) 26–32
7. Knuplesch, D., Reichert, M., Mangler, J., Rinderle-Ma, S., Fdhila, W.: Towards compliance of cross-organizational processes and their changes. In: BPM'12 Workshops. Volume 132 of LNBIP., Springer (2013) 649–661
8. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: Visual modeling of business process compliance rules with the support of multiple perspectives. In: ER'2013. Volume 8217 of LNCS., Springer (2013) 106–120
9. Semmelrodt, F., Knuplesch, D., Reichert, M.: Modeling the resource perspective of business process compliance rules with the extended compliance rule graph. In: BPMDS'14. Volume 175 of LNBIP., Springer (2014) 48–63
10. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: CoopIS'11. Volume 7044 of LNCS., Springer (2011) 82–99
11. Knuplesch, D., Reichert, M.: An operational semantics for the extended compliance rule graph language. Technical Report 2014-6, Ulm University (2014)
12. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: On the formal semantics of the extended compliance rule graph. Technical Report 2013-05, Ulm University (2013)

13. Ramezani, E., Fahland, D., van der Aalst, W.M.P.: Where did I misbehave? Diagnostic information in compliance checking. In: BPM'12. Volume 7481 of LNCS., Springer (2012) 262–278
14. Beck, H.: Automatisierte Überwachung von Business Process Compliance Regeln mit Daten-, Zeit-, Ressourcen- und Interaktions-Aspekten. Master Thesis, Ulm University, Germany (2014)
15. Becker, J., et al.: Generalizability and applicability of model-based business process compliance-checking approaches. BuR - Business Research **5**(2) (2012) 221–247
16. Fdhila, W., Rinderle-Ma, S., Knuplesch, D., Reichert, M.: Change and compliance in collaborative processes. In: SCC'15, IEEE (2015)
17. Alles, M., Kogan, A., Vasarhelyi, M.: Putting continuous auditing theory into practice: Lessons from two pilot implementations. Information Systems **22**(2) (2008) 195–214
18. Namiri, K., Stojanovic, N.: Pattern-Based design and validation of business process compliance. In: CoopIS'07. Volume 4803 of LNCS., Springer (2007) 59–76
19. van der Aalst, W.M.P., van Hee, K.M., van der Werf, J.M.E.M., Kumar, A., Verdonk, M.: Conceptual model for online auditing. Decision Support Systems **50**(3) (2011) 636–647
20. Accorsi, R.: An approach to data-driven detective internal controls for processaware information systems. In: DUMW'12. (2012) 29–33
21. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Eventbased monitoring of process execution violations. In: BPM'11. Volume 6896 of LNCS., Springer (2011) 182–198
22. Santos, E.A., Francisco, R., Vieira, A., de F.R. Loures, E., Busetti, M.: Modeling business rules for supervisory control of process-aware information systems. In: BPM'11 Workshops. Volume 100 of LNBIP., Springer (2012) 447–458
23. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: FASE'12. Volume 7212 of LNCS., Springer (2012) 146–162
24. Maggi, F., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: an approach based on colored automata. In: BPM'11. Volume 6896 of LNCS., Springer (2011) 132–147
25. Baumgrass, A., Baier, T., Mendling, J., Strembeck, M.: Conformance checking of RBAC policies in process-aware information systems. In: BPM'12 Workshops. Volume 100 of LNBIP., Springer (2012) 435–446
26. Outmazgin, N., Soffer, P.: A process mining-based analysis of business process work-arounds. Software & Systems Modeling (2014) 1–15
27. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: ICSOC'09. Volume 5900 of LNCS., Springer (2009) 500–515
28. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: ER'2010. Volume 6412 of LNCS., Springer (2010) 332–346
29. Knuplesch, D., Reichert, M., Fdhila, W., Rinderle-Ma, S.: On enabling compliance of cross-organizational business processes. In: BPM'13. Volume 8094 of LNCS., Springer (2013) 146–154
30. Knuplesch, D., Reichert, M., Pryss, R., Fdhila, W., Rinderle-Ma, S.: Ensuring compliance of distributed and collaborative workflows. In: CollaborateCom'13, IEEE (2013) 133–142

31. Turetken, O., Elgammal, A., van den Heuvel, W.J., Papazoglou, M.: Capturing compliance requirements: A pattern-based approach. IEEE Software **29**(3) (2012) 29–36
32. Sunkle, S., Kholkar, D., Kulkarni, V.: Toward better mapping between regulations and operational details of enterprises using vocabularies and semantic similarity. In: CAiSE'15 Forum, CEUR-WS 229–236
33. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: DECLARE: full support for loosely-structured processes. In: EDOC'07, IEEE (2007) 287–300
34. Hildebrandt, T., Mukkamala, R., Slaats, T.: Nested dynamic condition response graphs. In: FSEN'12. Volume 7141 of LNCS., Springer (2012) 343–350
35. Liu, R., Vaculín, R., Shan, Z., Nigam, A., Wu, F.: Business artifact-centric modeling for real-time performance monitoring. In: BPM'11. Volume 6896 of LNCS., Springer (2011) 265–280
36. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the event calculus. ACM Transactions on Intelligent Systems and Technology **5**(1) (2014) 17:1–17:30
37. Bragaglia, S., Chesani, F., Mello, P., Montali, M., Sottara, D.: Fuzzy conformance checking of observed behaviour with expectations. In: AI*IA'11. Volume 6934 of LNCS., Springer (2011) 80–91