An Operational Semantics for the Extended Compliance Rule Graph Language

David Knuplesch and Manfred Reichert

Institute of Databases and Information Systems,
Ulm University, Germany

david.knuplesch@uni-ulm.de
manfred.reichert@uni-ulm.de

Abstract

A challenge for any enterprise is to ensure conformance of its business processes with imposed compliance rules. Usually, the latter may constrain multiple perspectives of a business process, including control flow, data, time, resources, and interactions with business partners. Like for process modeling, intuitive visual languages have been proposed for specifying compliance rules. However, business process compliance cannot completely be decided at design time, but needs to be monitored during run time as well. In previous work we introduced the extended Compliance Rule Graph (eCRG) language that enables the visual monitoring of business process compliance regarding the control flow, data, time, and resource perspectives as well as the interactions a process has with business partners. This technical report introduces an operational semantics of the eCRG language. In particular, the state of a visual compliance rule is reflected through markings and annotations of an eCRG. The proposed operational semantics not only allows detecting compliance violations at run-time, but visually highlights their causes as well. Finally, it allows providing recommendations to users in order to proactively ensure for a compliant continuation of a running business process.

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.

1 Introduction

The correctness of business process models has been intensively discussed in literature for more than a decade [1–3]. While earlier 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 last years [4, 5]. Usually, *compliance rules* stem from domain-specific constraints, e.g., referring to corporate standards, legal regulations or evidenced best practices [6, 7] , and need to be ensured in all phases of the process life cycle [8, 9].

Approaches dealing with the compliance of business processes during their execution are covered by the notion of *compliance monitoring*. In order to detect and report run-time violations of compliance rules, events of running business process instances need to be considered (cf. Fig. 1). Note that two kinds of monitoring need to be distinguished: reactive and proactive monitoring. Regarding reactive monitoring, the process-aware information system only reports a compliance violation once it has occurred. In turn, proactive monitoring aims to proactively prevent potential compliance violations that might occur during the further course of process execution; e.g. by suggesting appropriate tasks that still need to be executed to meet the compliance rule.

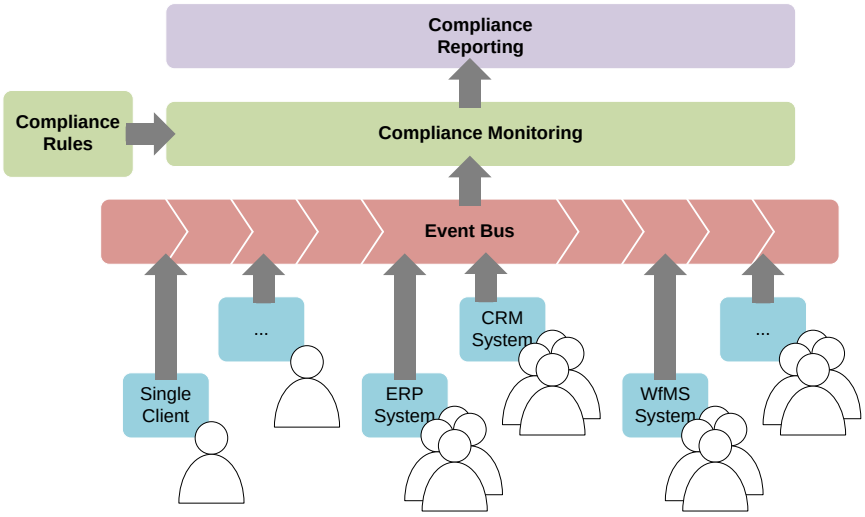


Fig. 1: Compliance monitoring [10, 5]

A multitude of approaches focusing on compliance monitoring at run-time were introduced during the last years [11–13]. While early suggestions focused on the control flow perspective, later proposals for monitoring compliance considered additional process perspectives as well [14, 15]. In particular, the data, resource, and time perspectives have been addressed. Other approaches, in turn, have focused on the traceability of compliance violations [10, 16]. Furthermore, [5] proposed 10 fundamental *compliance monitoring functionalities* (CMFs) that may be used to compare existing approaches for monitoring business process compliance. In this context, the authors stated that existing approaches do not provide a satisfactory solution that combines an expressive language with full traceability capabilities [5].

To close this gap, this report introduces an operational semantics for the extended Compliance Rule Graph (eCRG) language. The latter has been introduced in [17, 18] and enables

the visual monitoring of business process compliance. In particular, the operational semantics annotates eCRGs with text, colors and symbols. In order to deal with compliance rules, which are triggered multiple times during the execution of a business process instance, the operational semantic creates and annotates multiple instances of an eCRG in parallel. The annotated instances of an eCRG not only indicate compliance violations, but may be also utilized for recommending the next process steps (i.e. activities), whose execution will ensure compliance. Furthermore, they provide a suitable basis for compliance metrics. Note that the eCRG language is a powerful visual notation for compliance rules that adequately supports the control flow, data, time, and resource perspectives as well as interactions with business partners. Consequently, this report not only provides the operational basis for monitoring control flow constraints, but enables the monitoring of the latter perspectives as well. Altogether the provided operational semantics provides the basis for

- utilizing the eCRG language for monitoring business process compliance during process execution.
- monitoring the conformance with compliance rules related to any process perspective; i.e., the control flow, data, time, resource perspectives as well as the interactions a process has with business partners.
- dealing with compliance rules, which are are triggered multiple times.
- reasoning about the violation of compliance rules – reactively & proactively.
- specifying compliance metrics and measures.

The remainder of this report is structured as follows: Section 2 discusses related work. Section 3 provides a motivating example. Backgrounds on the *extended Compliance Rule Graph* (eCRG) language are introduced in Section 4. Section 5 provides an operational semantics for the eCRG language. In particular this operational semantics formally specifies the transitions between the states of compliance rules. In turn, Section 6 evaluates and classifies the compliance of an eCRG. Furthermore, the specification of compliance metrics and measures is introduced as well as the provision of recommendations to users. Finally, Section 7 concludes the paper and provides an outlook on future research.

2 Related Work

For a decade, business process compliance has increasingly gained attention and several surveys have been published in the meantime (e.g., [19, 8, 20, 21, 5]). Along this trend, interest in *compliance monitoring* and *continous auditing* [12] has grown as well. For example, [11] enriches process models with a semantic layer of internal controls. In [13, 22], in turn, the detailed architectures of an online auditing tools (OLAT) are described. An OLAT allows monitoring the operations of an organization in detective, corrective and preventive modes. The spectrum of techniques applied for compliance monitoring is wide spread: *Behavioural profiles* [23] utilize ordering relations in this context, whereas *Supeverisory Control Theory* [24] prevents users from performing actions leading to compliance violations. Furthermore, visual declarative constraints [25], which are transformed into *Event Calculus* and *Linear Temporal Logic* (LTL), have shown increasing popularity. *Fuzzy conformance checking* [26] calculates an evaluation score that indicates how much the observed process instance complies instead of providing a simple *yes/no* answer.

In order to enable fine-grained compliance diagnostics at run-time, *Compliance Rule Graphs* [10] and colored automata [16] have been suggested. However, both mainly focus on control flow perspective. In turn, [5] compares approaches for monitoring business process compliance based on 10 compliance monitoring functionalities (CMF). In particular, it is emphasized that existing approaches do not provide a satisfactorily solution that combines an expressive language (CMF 1-5) with full compliance traceability (CMF 8+9). Table 1 summarizes the results from [5].

Table 1: Compliance monitoring functionalities [5]

Approach	CMF 1 time	CMF 2 data	CMF 3 resources	CMF 4 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 [16]	+/-	-	-	+	-	-	+	+	+	+/-
Mubicon EC [25, 15]	+	+/-	+	+	+	+	+	-	+/-	+/-
ECE Rules [26]	+	+/-	+	+	-	-	+	-	+/-	+
SCT [24]	+/-	-	+	+	+	-	-	+	-	-
SeaFlows [10]	+/-	+/-	+/-	+	+/-	+	+	+	+	+/-

As opposed to the approaches described above, [27] monitors performance measures in the context of artifact-centric process models. In turn, [28, 14, 29] provide techniques to *a posteriori* verify the compliance of execution logs with a set of constraints. Some of these approaches not only focus on the control flow perspective, but take the time perspective [14] or resource perspective [28] into account as well.

A priori or *design time* compliance checking has been addressed by a multitude of approaches for a long time. Most of them apply model checking techniques (e.g., [30–33]). In addition, some of these design time approaches use visual compliance rules and not only consider the control flow perspective, but the data, interaction or time perspectives as well. Other compliance checking approaches, in turn, are based on *Petri-Nets* [34] and Mixed-Integer Programming [35].

Finally, there are few frameworks, which address the integration of business process compliance throughout the entire process lifecycle [9, 36, 8, 37, 38].

3 Motivating Example

This section introduces a motivating example, which refers to the event log from Fig. 2 and deals with an order-to-delivery process. The latter is subject to three compliance rules, which stem from internal guidelines.

Note that compliance rule c_1 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. 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). Besides the violation of c_1, compliance rule c_2 is violated twice as well. While the violated instance of rule c_1 will never be successfully completed, the violations of c_2 still may be healed by informing the *agent*, who sent the requests, about the results of the approvals.

The compliance rule examples further indicate that solely monitoring control flow dependencies between tasks is not sufficient in order to ensure compliance at run-time. In addition, constraints in respect to the data, time and resource perspectives of a business process as well as the interactions this process has with partner processes must be monitored [39, 21, 17, 18]. 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) represents an interaction with a business partner. The phrase "by different staff members" deals with the resource perspective, whereas the condition "at max three days" refers to the time perspective. To meet practical demand, compliance monitoring must not abstract from these process perspectives.

#	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€
39	**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 then ***10,000*** is ***received from an agent***, the request must not be approved unless the solvency of the respective customer was checked. The latter task must be started ***at max three days*** after the receipt. Further, 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 days***.

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

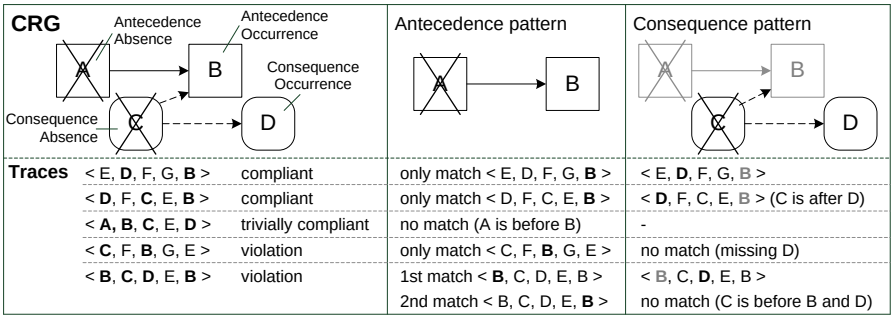
4 Fundamentals of Extended Compliance Rule Graphs

This paper utilizes the extended Compliance Rule Graph (eCRG) language for compliance monitoring. Since this language is based on the Compliance Rule Graph (CRG) language, we first introduce CRGs before presenting eCRG fundamentals.

4.1 Compliance Rule Graphs

The Compliance Rule Graph (CRG) language was introduced in [40, 10, 41]. It allows for the visual modeling of compliance rules that focus on the control flow perspective (i.e., sequence flow) of business processes. A CRG corresponds to an acyclic graph that consists of an *antecedence pattern* as well as one or multiple related *consequence patterns*. Both kinds of patterns are modeled using *occurrence* and *absence nodes*, which either express the occurrence or absence of events (e.g. events related to the execution of a particular task). Furthermore, the edges connecting these nodes express control flow dependencies.

As illustrated in Fig. 3, an event trace (i.e., a finite sequence of events related to the same process instance) is considered as *compliant* with a CRG iff for each match of the antecedence pattern there is at least one corresponding match for one of the consequence pattern. In turn, a trace is considered as *trivially compliant* iff there is no match of the antecedence pattern at all. As example consider the CRG from Fig. 3. It expresses that for each B not preceded by an A, a D must occur. Further, there must be no C that precedes B and D.



Traces			Antecedence pattern	Consequence pattern
	< E, **D**, F, G, **B** >	compliant	only match < E, D, F, G, **B** >	< E, **D**, F, G, B >
	< **D**, F, **C**, E, **B** >	compliant	only match < D, F, C, E, **B** >	< **D**, F, C, E, B > (C is after D)
	< **A, B**, **C**, E, **D** >	trivially compliant	no match (A is before B)	-
	< **C**, F, **B**, G, E >	violation	only match < C, F, **B**, G, E >	no match (missing D)
	< **B**, **C**, **D**, E, **B** >	violation	1st match < **B**, C, D, E, B >	< B, C, **D**, E, B >
			2nd match < B, C, D, E, **B** >	no match (C is before B and D)

Fig. 3: CRG example and semantics [8]

4.2 Extended Compliance Rule Graph

The CRG language focuses on the *control flow perspective* of compliance rules, but ignores other perspectives. In [17, 18], therefore, we introduced the extended Compliance Rule Graph (eCRG) as a visual language for modeling compliance rules not only covering the control flow perspective, but providing integrated support for the resource, data and time perspectives as well as for interactions with business partners. To cover these various perspectives, the eCRG language allows for *attachments* in addition to *nodes* and *connectors* (i.e. edges). Thereby, nodes refer to entities (e.g. a data object) or events, whereas edges and attachments are used to refine the nodes or edges they are affiliated to. Furthermore, an eCRG may contain *instance nodes* referring to particular objects, which exist independently from the respective rule (e.g. *Mr. Smith, postnatal ward, physician*). Hence, instance nodes are neither part of

the antecedence nor the consequence pattern, but constitute the *instance patter*. Fig. 4 gives an overview of the elements of the eCRG language.

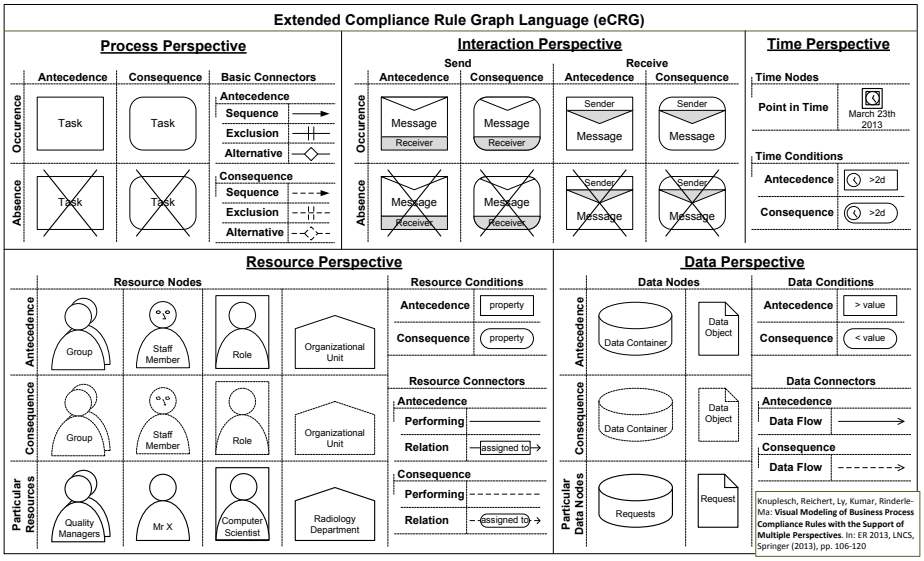


Fig. 4: Elements of the eCRG language [17, 18]

The elements of the eCRG language are partitioned into the control flow, data, time, resource and interaction perspectives that will be described in the following in more detail (cf. Fig. 4).

Control flow perspective. Modeling the control flow perspective of compliance rules is supported through four kinds of *task nodes*, i.e., antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence task nodes. Based on these nodes it can be expressed whether or not particular tasks shall be executed. In addition, two kinds of *sequence flow connectors* are provided that allow constraining the execution sequence of tasks. Note that the absence of a sequence flow indicates parallel flow. Furthermore, *exclusive connectors* express mutual exclusion of the tasks they refer to. Finally, *alternative connectors* express that at least one of the connected tasks must occur.

Interaction perspective. The interaction perspective supports the exchange of messages with business partners. According to task nodes, four kinds of sending and four kinds of receiving *message nodes* are provided, i.e., antecedence occurrence, antecedence absence, consequence occurrence, and consequence absence nodes.

Time perspective. The eCRG language offers the following elements for modeling the time perspective: *Point-in-time nodes*, *time condition attachments*, and *time distance connectors*. *Point-in-time nodes* express a particular date or point-in-time (e.g. 26th October 2014). *Time conditions*, in turn, may be attached to task nodes and sequence flow connectors to constrain the duration of a task or the time distance between *task nodes*, *message nodes* and *point-in-time nodes*. Finally, *time distance connectors* allow constraining the time distance without implying a particular sequence.

Data perspective. *Data container nodes* and *data object nodes* support the modeling of the data perspective in eCRGs. Furthermore, *data flow connectors*, *data relation connectors* and *data condition attachments* are provided. *Data container nodes* refer to process data elements or global data stores. By contrast, *data object nodes* refer to particular data values and data object instances. Both kinds of data nodes may be part of the antecedence or consequence pattern, or represent a particular data container and data object, respectively. *Data flow connectors* define which process tasks read or write which data objects or data containers. To constrain data containers, data objects and data flow, *data conditions* may be attached. In turn, *data relation connectors* allow comparing data objects.

Resource perspective. For modeling the resource perspective of compliance rules, *resource nodes* are provided, i.e., *staff member*, *role*, *group*, and *organizational unit nodes*. Similar to task nodes, *resource nodes* may be part of the antecedence or consequence pattern. Alternatively, they may represent a particular resource instance (e.g. *Mr. Smith, postnatal ward, physician*). To specify dependencies among resources, *resource relation connectors* are provided. In turn, *resource condition attachments* constrain a particular resource node. Finally, the *performing relation* indicates the performer of a task node.

Fig. 5 applies the eCRG language in order to model the compliance rules from our motivating example in Section 3, which have been presented in verbalized form in Fig. 2. In particular, Fig. 2(c_1) addresses all process perspectives, i.e., the control flow, data, time and resource perspectives as well as interactions with business partners. In turn, Fig. 2(c_2) does not refer to the resource perspective, whereas time and interaction perspectives are not addressed in Fig. 2(c_3).

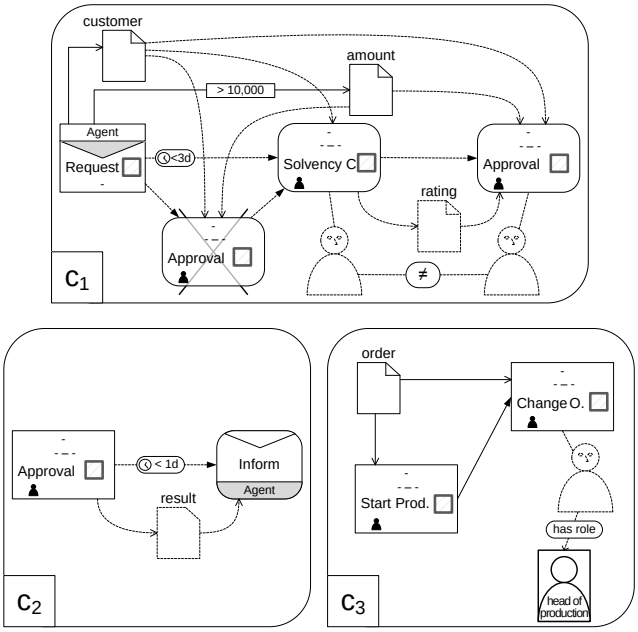


Fig. 5: Modeling compliance rules c_{1-3} with the eCRG language

A formal specification of an eCRG is provided in Def. 1 (for a more detailed definition of eCRGs including a formal definition of ϕ see [42]).

Definition 1 (Extended Compliance Rule Graph (eCRG)).
Let $\mathfrak{T}$ be the set of points in time, $\mathcal{R}$ be the set of human resources, and Ω be the set of all data objects. Then: An extended Compliance Rule Graph (eCRG) is a tuple $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ with

- $N := T \cup MN \cup O \cup C \cup R \cup P$ *being the set of* **nodes** *that may be partitioned into the sets of* **task nodes** T, **message nodes** MN, **data object nodes** O, **data container nodes** C, **resource nodes** R, *and* **point-in-time nodes** $P \subset \mathfrak{T}$.
- $E := \overrightarrow{sf} \cup \overrightarrow{df} \cup \overrightarrow{pfm} \cup \overrightarrow{rr}$ *being the set of* **edges** *that may be partitioned into the sets of* **sequence flow edges** $\overrightarrow{sf}$, **data flow edges** $\overrightarrow{df}$, **performing relations** $\overrightarrow{pfm}$, *and* **resource relations** $\overrightarrow{rr}$.
- $\diamond := {}^{\diamond}dc \cup {}^{\diamond}tc \cup {}^{\diamond}rr$ *being the set of* **attachments** *that may be partitioned into the sets of* **data conditions** ${}^{\diamond}dc$, **time conditions** ${}^{\diamond}tc$, *and* **resource conditions** ${}^{\diamond}rc$,
- $type : T \cup MN \cup \overrightarrow{df} \rightarrow \mathcal{T}$ *mapping each task node (message node, data flow edge) to a* **task type** *(***message type**, **parameter name**),
- $src : E \rightarrow N$ *($tgt : E \rightarrow N$) mapping each edge to its* **source** *(***target**) **node**,
- $at : \diamond \rightarrow N \cup E$ *mapping each attachment to the* **underlying node or edge** *it is attached to, and*
- $pt : N \cup E \cup \diamond \rightarrow \{AO, AA, CO, CA, I\}$ *mapping each element of an eCRG to the* **corresponding pattern**.

Further:

- $\Lambda_\Psi := N \cup E \cup \diamond$ *is the set of* **all elements**,
- $\Gamma_\Psi := T \cup MN$ *is the set of* **task and message nodes**,
- $\phi_\Phi : \Lambda_\Psi \rightarrow \Lambda_\Psi$ *maps each element to its* **affiliation**; *i.e., the element to which it is affiliated*[3].
- *For each set $X \subset \Lambda_\Psi$ of elements of an eCRG and each pattern $y \in \{AO, AA, CO, CA, I\}$, we define $X^y := \{x \in X | pt(x) = y\}$ as the* **pattern y of X**.

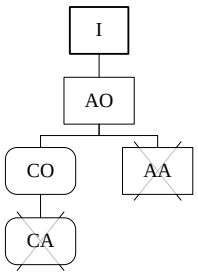


Fig. 6: Dependencies of eCRG pattern

Function pt partitions the elements of an eCRG in five patterns, which are the *instance pattern* (I) as well as the a*ntecedence occurrence* (AO), *antecedence absence* (AA), *consequence occurrence* (CO), and *consequence absence* (CA) patterns. Dependencies among these patterns are shown in Fig. 6, whereas the lower ones depend on the upper ones they

are connected to. Based on the latter, Def. 2 formally specifies how connected elements constrain and depend on each other. In particular, a node, edge or attachment λ_1 depends on another element λ_2, if they are connected and the pattern of λ_1 depends on the one of λ_2. For example, task node *production* depends on its outgoing sequence flow edge in Fig. 5(c_3). In turn, in Fig. 5(c_3), message node *request* does not depend on both outgoing sequence flows. Instead, the latter depend on the message node.

Definition 2 (Pattern Dependency Order).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and let $\lambda_1, \lambda_2 \in \Lambda_\Psi$ be two elements of the eCRG. Then:

- $\blacktriangleright$ *defines the partial dependency order over the set of patterns as specified in Fig. 6 (e.g. $I \blacktriangleright AA$),*
- *We say λ_1 has a higher dependency level than λ (i.e., $\lambda_1 \rhd \lambda_2$), iff the pattern of λ_1 is greater than the one of λ_2 according to the dependency order $\blacktriangleright$, i.e.,*

$$\lambda_1 \rhd \lambda_2 :\Leftrightarrow pat(\lambda_1) \blacktriangleright pat(\lambda_2)$$

- *and $\trianglerighteq$, $\triangleq$, $\lhd$ and $\trianglelefteq$ are defined accordingly.*

5 eCRG Operational Semantics

This section introduces the operational semantics of the eCRG language that enables visually monitoring business process compliance at run-time. As discussed in Section 1, the latter is based on event streams that occur during the execution of business processes. In particular, compliance monitoring shall detect or, if possible, prevent compliance violations. For this purpose, first of all, Section 5.1 introduces the different events that are supported. Second, the fundamental states of a compliance rule are introduced in Section 5.2. Third, Section 5.3 specifies markings that annotate the elements of an eCRG with text, colors and symbols. Finally, Section 5.4 specifies the processing of events and discusses how the latter evolve and unfold markings of an eCRG (cf. Section 5.4).

5.1 Events

As this report addresses compliance monitoring in respect to multiple process perspectives, we not only monitor events that refer to the start and end of tasks. In addition, we consider events that correspond to the sending and receiving of messages as well as data flow events that log how activities read from and write to data sources. Furthermore, events may include temporal information as well as information about involved resources.

Table 2 summarizes the event types supported by our approach. Note that each event includes the time it occurs as well as a unique id. The latter enables us to identify correlations between the start, end and data flow events related to the same task or message.

Table 2: Supported Events

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

Based on the events from Table 2, Def. 3 formally specifies event logs and streams.

Definition 3 (Event Log or Event Stream).
Let $\mathcal{E}$ be the set of events (cf. Table 2) and let $\mathfrak{T}$ be the discrete set of points in time. Then:

- $\mathcal{L} : \mathbb{N} \to \mathcal{E} \cup \{\varnothing\}$ *is an* **event stream** *or* **event log**,
- $time : \mathcal{E} \to \mathfrak{T} : time(event(\vartheta, \dots)) := \vartheta$ *maps each event to the* **corresponding point-in-time**.

Note that we assume that event streams are correct; i.e., they do not deviate (cf. [43]) from the real process, ids are unique, and events are provided in an ascending order (i.e., $\forall i, j \in [1..pos_{\mathcal{L}}] : i < j \Rightarrow time(\mathcal{L}(i)) \leq time(\mathcal{L}(j))$).

5.2 States of Compliance Rules

When monitoring the compliance of running process instances, compliance rules take varying states [10, 8]. Fig. 7 outlines the states that are supported by our approach. The most

fundamental state is NOT_ACTIVATED, i.e., the compliance rule does not concern the running process instances until now. The opposite state ACTIVATED means that the compliance rule affects the process instance and includes the sub-states TEMPSATISFIED and TEMPVIOLATED. TEMPSATISFIED is further partitioned into VIOLABLE and SATISFIED, whereas TEMPVIOLATED includes the sub-states PENDING and VIOLATED. As explained by our motivating example in Section 3, business process can multiply trigger (i.e. activate) compliance rule. Hence, a compliance rule can be in state ACTIVATED multiple times as indicated by superscript "+". Note that each of these *activations* of a compliance rule can take a different sub-state. For instance, the event log of our motivating example activates compliance rule c_1 twice (cf. Fig. 2). The first activation is SATISFIED, whereas the second activation remains in state VIOLATED.

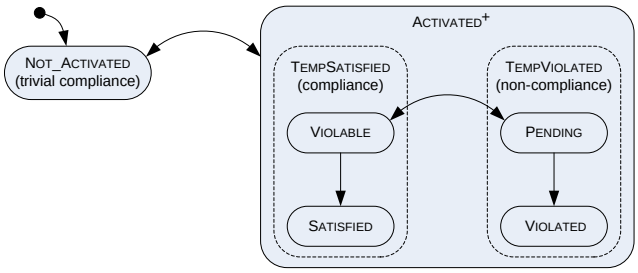


Fig. 7: States of compliance rules

5.3 eCRG Markings

To monitor the state of a compliance rule, we annotate and mark the elements of an eCRG (cf. Section 4, [17, 18]) with symbols, colors and text (cf. Figs. 8). Such a *marking of an eCRG* (i.e., annotated eCRG) highlights whether or not the events corresponding to a node have occurred so far. Further, it describes whether conditions corresponding to edges and attachments are satisfied, violated, or still will be evaluated (cf. Figs. 8). Since there may be different activations and instantiations of a particular compliance rule, Def. 4 defines the *state of an eCRG* in terms of a set of markings, which, in turn, can be utilized to calculate the corresponding states of compliance as introduced in Section 5.2.

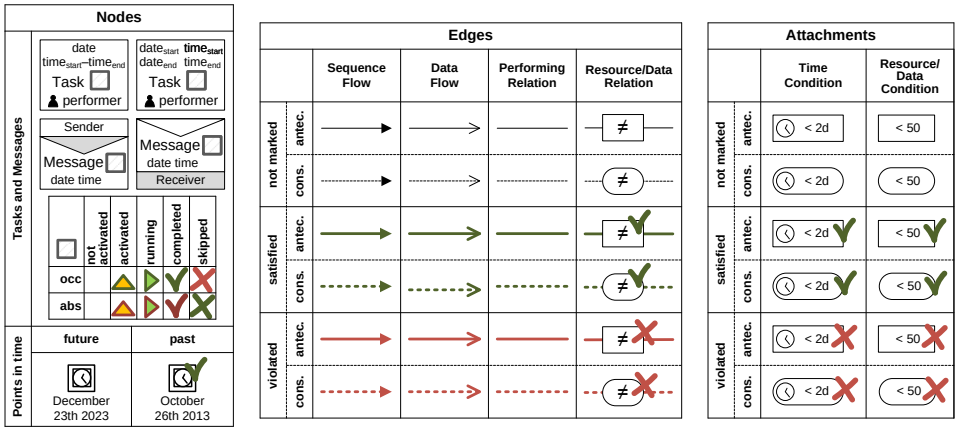


Fig. 8: Annotations of eCRG elements

Definition 4 (Markings and State of an eCRG).
Let $\mathcal{I}$ be a set of unique identifiers. Let further $\mathfrak{T}$ be the set of points in time, $\mathcal{R}$ be the set of human resources, and Ω be the set of all data objects, whereas ϵ is the empty value, i.e., the placeholder for not yet set identifiers, points in time, human resources, and data objects. Furthermore, let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG. Then: A marking M of Ψ is a tuple $M = (m, res, val, id, t_s, t_e)$ where

- $m : \Lambda_\Psi \rightarrow \{\square, \triangle, \blacktriangleright, \checkmark, \times\}$ **marks** *each element of the eCRG with either* Not_Activated/Not_Marked *(□), Activated (△), Running (▶), Completed/ Satisfied (✓), or Skipped (×).*
- $res : T \cup R \cup \overrightarrow{pfm} \rightarrow \{\epsilon\} \cup \mathcal{R}$ *assigns a* **resource** *to each task node, resource node and performing relation edge,*
- $val : O \cup \overrightarrow{df} \rightarrow \{\epsilon\} \cup \Omega$ *assigns a* **value** *to each data object node and data flow edge,*
- $id : T \cup MN \cup C \cup \overrightarrow{df} \rightarrow \{\epsilon\} \cup \mathcal{I}$ *assigns an unique* **identifier** *to each task node, message node, data container node, or data flow edge,*
- $t_s : T \cup MN \cup P \rightarrow \{\epsilon\} \cup \mathfrak{T}$ *(and $t_e : T \cup MN \cup P \rightarrow \{\epsilon\} \cup \mathfrak{T}$ respectively) assigns a* **starting** *(***ending***)* **time** *to each task node, message node, and point-in-time node.*

Further, we define

- $M_\Psi^{in} := (m, res, val, id, t_s, t_e)$ *as the* **initial marking** *of eCRG Ψ, where*
  - $m(x) := \square$ *marks every element x with □,*
  - $t_s(x) := t_e(x) := \epsilon$ *sets starting and ending times to ϵ, and*
  - $res(x) := \begin{cases} \epsilon, \text{iff } x \notin I \\ x, \text{else} \end{cases}$ $val(x) := \begin{cases} \epsilon, \text{iff } x \notin I \\ x, \text{else} \end{cases}$ $id(x) := \begin{cases} \epsilon, \text{iff } x \notin I \\ x, \text{else} \end{cases}$
    only assign resources, values and identifiers to elements of the instance pattern.
- $\mathfrak{M}_\Psi$ *as the set of* **all markings** *of Ψ,*
- $\mathcal{M}_\Psi^{\mathcal{L},i} \subseteq \mathfrak{M}_\Psi$ *as the* **state of eCRG** *Ψ after processing event stream $\mathcal{L}$ until entry $i \in \mathbb{N}$, and*
- $\mathcal{M}_\Psi^{in} := \{M_\Psi^{in}\}$ *as the* **initial state** *of Ψ.*

5.4 Operational Semantics

In order to enable the processing of events in the context of the extended Compliance Rule Graph (eCRG) language, this section introduces the eCRG operational semantics. In particular Defs. 5-10 specify how the different event types are processed (cf. Fig. 9). First, all markings must be updated to the point-in-time of the event occured (cf. Def. 5). Second, effects of the update (i.e., the updated annotations) are propagated to adjacent elements (cf. Def. 10). Third, Defs. 6-9 specify the actual *handling* of an event based on its type. Finally, annotations are propagated once again before completing the processing of an event.
Note that the first two steps may be skipped if time does not differ from the one of the event directly processed before. In turn, these two steps may be applied without the subsequent ones in order to calculate the current state of a compliance rule at any point-in-time between two events.

5.4.1 Update Marking

Def. 5 specifies the update of a marking to the current point-in-time. In particular, the annotation of point-in-time nodes changes from Not_Marked □ to Completed ✓, if the

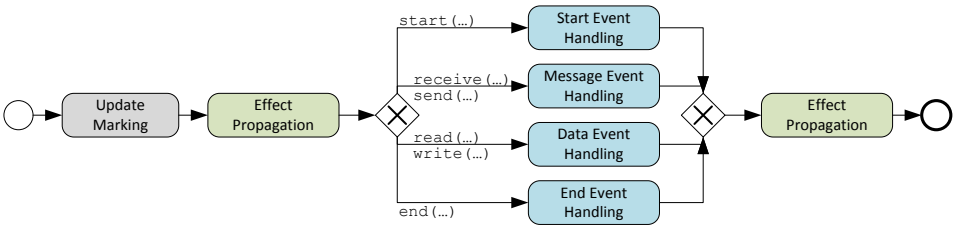


Fig. 9: Processing of start, message, data and end events

points in time to which the nodes refer have passed (t1). Furthermore, time conditions on running task nodes or sequence flow edges will be SKIPPED (×) if they are no longer satisfiable (t2).

Definition 5 (Update Marking).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ *and let* $\vartheta \in \mathfrak{T}$ *be a point-in-time. Then:* $upd^\vartheta : \mathfrak{M}_\Psi \to \mathfrak{M}_\Psi$ *with* $upd^\vartheta(M) := (m', res, val, id, t_s, t_e)$ *calculates the* **update** *of marking* M *to point-in-time* ϑ*, where*

$$m'(\lambda) := \begin{cases} \checkmark, \ \textit{if } \lambda \in P & \wedge m(\lambda) = \square \wedge \lambda \leq \vartheta, & \textit{(t1)} \\ \times, \ \textit{if } \lambda \in {}^\diamond tc & \wedge m(\lambda) = \square \wedge (t := at({}^\diamond tc) \in T) & \textit{(t2)} \\ & \wedge \left(\forall \varsigma \in \mathfrak{T} : (\varsigma \geq \vartheta) \Rightarrow \neg {}^\diamond tc(t_s(t), \varsigma)\right), & \\ \times, \ \textit{if } \lambda \in {}^\diamond tc & \wedge m(\lambda) = \square \wedge \left((n_1, n_2) := at({}^\diamond tc) \in \overrightarrow{sf}\right) & \textit{(t2)} \\ & \wedge \left(\forall \varsigma \in \mathfrak{T} : (\varsigma \geq \vartheta) \Rightarrow \neg {}^\diamond tc(t_e(n_1), \varsigma)\right), & \\ m(\lambda), \ \textit{else} & & \end{cases}$$

Fig. 10 updates marking B of compliance rule c_1 from our motivating example (cf. Fig. 2) to the point in time *4/7/2013 16:05*. According to Def. 5(t2), the time condition on the sequence flow edge from message node is marked as skipped, because it can not be satisfied any more.

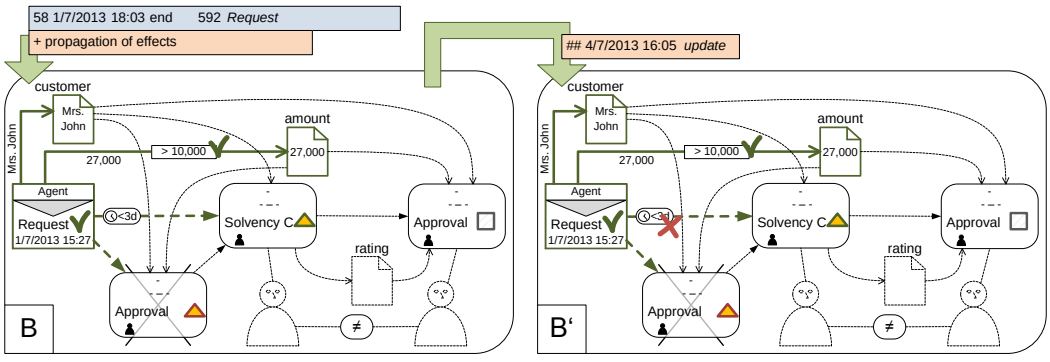


Fig. 10: Update of markings

5.4.2 Start and Message Event Handling

Def. 6 (Def. 7) non-deterministically handles start events (message events), in order to deal with different instantiations of a compliance rule. In particular, the set of matching and activated task nodes (message nodes) is determined first. Then, for each subset a new marking is instantiated, which solely changes the marking of the selected nodes from ACTIVATED ($\triangle$) to RUNNING ($\blacktriangleright$), and, accordingly, sets the identifiers, resources and starting times (cf. Fig. 11). Node that the empty set is among the subsets and the result hence includes the unchanged original marking as well.

Definition 6 (Start Event Handling).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ *and* $\sigma = \texttt{start}(\vartheta, \iota, \tau, \rho)$ *be a start event. Then:*

- $\Delta^\sigma(M) := \{\lambda \in T | m(\lambda) = \triangle \wedge type(\lambda) = \tau\}$ *is the set of* **matching** *and activated* **task nodes**,
- *For each subset* $\delta \subseteq \Delta^\sigma(M)$, $hdl^\sigma_\delta : \mathfrak{M}_\Psi \to \mathfrak{M}_\Psi$ *with* $hdl^\sigma_\delta(M) := (m^\delta, res^\delta, val, id^\delta, t^\delta_s, t_e)$ *calculates the* **handling of start event** σ *by node set* δ *and marking* M*:*

$$m^\delta(\lambda) := \begin{cases} \blacktriangleright, & \textit{iff } \lambda \in \delta \\ m(\lambda), & \textit{else} \end{cases} \qquad res^\delta(\lambda) := \begin{cases} \rho, & \textit{iff } \lambda \in \delta \\ res(\lambda), & \textit{else} \end{cases}$$
$$id^\delta(\lambda) := \begin{cases} \iota, & \textit{iff } \lambda \in \delta \\ id(\lambda), & \textit{else} \end{cases} \qquad t^\delta_s(\lambda) := \begin{cases} \vartheta, & \textit{iff } \lambda \in \delta \\ id(\lambda), & \textit{else} \end{cases}$$

- $hdl^\sigma : \mathfrak{M}_\Psi \to 2^{\mathfrak{M}_\Psi}$ *with* $hdl^\sigma(M) := \{hdl^\sigma_\delta(M) | \delta \in \Delta^\sigma(M)\}$ *calculates all handlings of* σ *by* M.

Definition 7 (Message Event Handling).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ *and let* $\sigma = \texttt{send}(\vartheta, \iota, \tau)$ *(and* $\texttt{receive}(\vartheta, \iota, \tau)$ *respectively) be a message event. Then:*

- $\Delta^\sigma(M) := \{\lambda \in MN | m(\lambda) = \triangle \wedge type(\lambda) = \tau\}$ *is the set of* **matching** *and activated* **message nodes**,
- *For each subset* $\delta \subseteq \Delta^\sigma(M)$, $hdl^\sigma_\delta : \mathfrak{M}_\Psi \to \mathfrak{M}_\Psi$ *with* $hdl^\sigma_\delta(M) := (m^\delta, res^\delta, val, id^\delta, t^\delta_s, t_e)$ *calculates the* **handling of message event** σ *by node set* δ *and marking* M*:*

$$m^\delta(\lambda) := \begin{cases} \blacktriangleright, & \textit{iff } \lambda \in \delta \\ m(\lambda), & \textit{else} \end{cases} \quad id^\delta(\lambda) := \begin{cases} \iota, & \textit{iff } \lambda \in \delta \\ id(\lambda), & \textit{else} \end{cases} \quad t^\delta_s(\lambda) := \begin{cases} \vartheta, & \textit{iff } \lambda \in \delta \\ id(\lambda), & \textit{else} \end{cases}$$

- $hdl^\sigma : \mathfrak{M}_\Psi \to 2^{\mathfrak{M}_\Psi}$ *with* $hdl^\sigma(M) := \{hdl^\sigma_\delta(M) | \delta \in \Delta^\sigma(M)\}$ *calculates all handlings of* σ *by* M.

Fig. 11 illustrates the handling of a start and message events. In particular, the receipt of the message *request* starts the corresponding message node of marking 0 in the upper example; i.e., the node is marked as RUNNING ($\blacktriangleright$) and labeled with the receive time of the message. The lower example handles the start-event of task *solvency check*. The corresponding task node is marked as RUNNING. Furthermore, its the start time and performer of the task are specified.

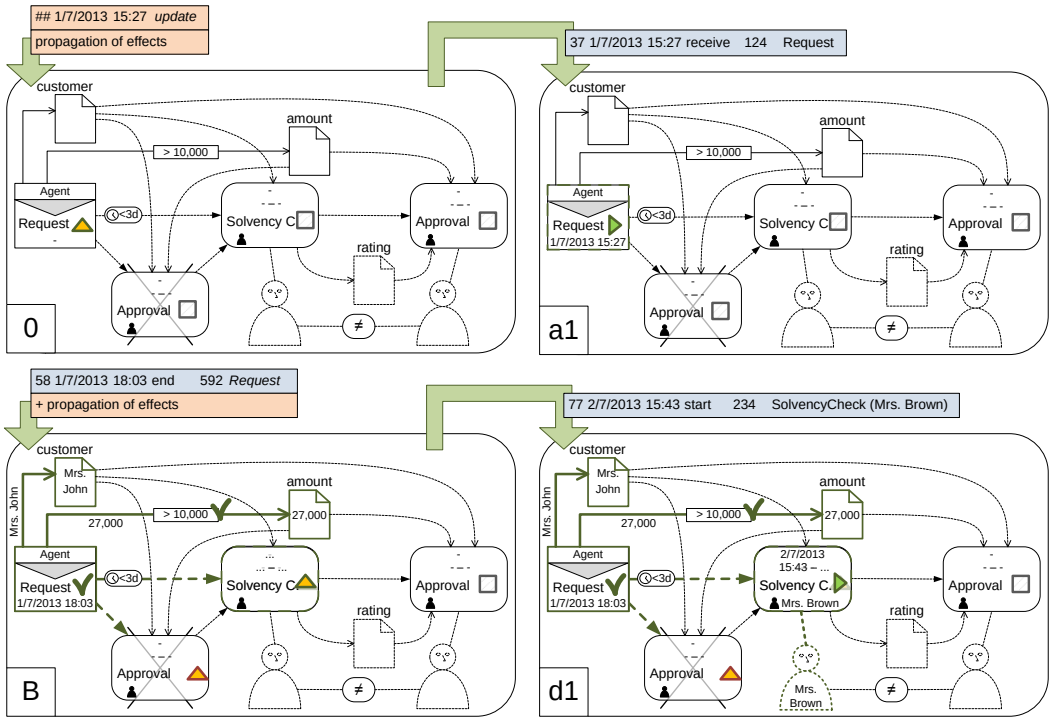


Fig. 11: Handling of start and message events

5.4.3 End Event Handling

Since we use unique identifiers for start, message and end events, the latter can be processed deterministically (cf. Def. 8). In particular, the annotations of all nodes assigned to the identifier of the end event and marked with RUNNING (▶), are changed to COMPLETED (✓) and their ending time is set accordingly (cf. Fig. 12).

Definition 8 (End Event Handling).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ *and* $\sigma = \textbf{\textit{end}}(\vartheta, \iota, \tau, \rho)$ *(and* $\textbf{\textit{end}}(\vartheta, \iota, \tau)$ *respectively) be an end event. Then:* $hdl^\sigma : \mathfrak{M}_\Psi \to 2^{\mathfrak{M}_\Psi}$ *with* $hdl^\sigma(M) := \{(m', res, val, id, t_s, t'_e)\}$ *calculates the the* **handling of end event** σ *by marking* M*:*

$$m' := \begin{cases} ✓, & \textit{iff } id(\lambda) = \iota \\ m(\lambda), & \textit{else} \end{cases} \qquad t'_e(\lambda) := \begin{cases} \vartheta, & \textit{iff } id(\lambda) = \iota \\ id(\lambda), & \textit{else} \end{cases}$$

Fig. 12 shows the handling of an end event. In particular, the processing of the message *request* is finished and the the corresponding message node of marking $a5$ is marked as COMPLETED (✓).

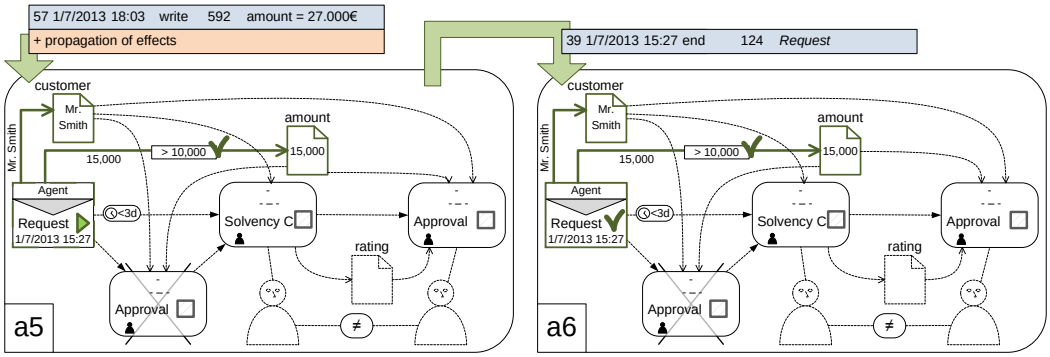


Fig. 12: Handling of end events

5.4.4 Data Event Handling

We can also process data events deterministically, since the combination of the used unique identifiers and the used parameter names clearly refers to the data flow edges concerned (cf. Def. 9). In particular, the latter is marked as SATISFIED (✓) and annotated with the data value and the identifier of the data source or data target; i.e., the identifier of a data container (cf. Fig. 13).

Definition 9 (Data Event Handling).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ*. Further, let* $\sigma = \texttt{write}(\vartheta, \iota, \upsilon \xrightarrow{par} \omega)$ *(and* $\texttt{read}(\vartheta, \iota, \upsilon \xleftarrow{par} \omega)$ *respectively) be a data event. Then:* $hdl^\sigma : \mathfrak{M}_\Psi \to 2^{\mathfrak{M}_\Psi}$*, with* $hdl^\sigma(M) := \{(m', res, val', id', t_s, t_e)\}$ *calculates the* **handling of data event** σ *by marking* M*:*

$$m'(\lambda) := \begin{cases} \checkmark, & \text{if } \lambda = (n,t) \in \overrightarrow{df} \wedge m(\lambda) = \square \wedge id(n) = \iota \wedge m(n) = \blacktriangleright \wedge type(\lambda) = par \\ m(\lambda), & \text{else} \end{cases}$$

$$val'(\lambda) := \begin{cases} \upsilon, & \text{if } \lambda = (n,t) \in \overrightarrow{df} \wedge m(\lambda) = \square \wedge id(n) = \iota \wedge m(n) = \blacktriangleright \wedge type(\lambda) = par \\ val(\lambda), & \text{else} \end{cases}$$

$$id'(\lambda) := \begin{cases} \omega, & \text{if } \lambda = (n,t) \in \overrightarrow{df} \wedge m(\lambda) = \square \wedge id(n) = \iota \wedge m(n) = \blacktriangleright \wedge type(\lambda) = par \\ id(\lambda), & \text{else} \end{cases}$$

Fig. 13 illustrates the handling of a data events. The writing data flow event *amount* of the message *request* is handled in the upper example. In particular, the corresponding outgoing data flow edge of message node *request* is marked as SATISFIED (✓) and annotated with the amount of *15.000*. The lower example handles the reading data flow event *customer* that marks the incoming data flow edge of task *solvency check* as SATISFIED (✓) and annotates it with the customer *Mr. Smith*. Note that the latter annotation *Mr. Smith* does not meet the value *Mrs. John* of the data object *customer*; i.e., the handling of data flow events allows for conflicts. Furthermore, note that we omit the identifiers of data sources (i.e., data containers) in our examples for the sake of simplicity.

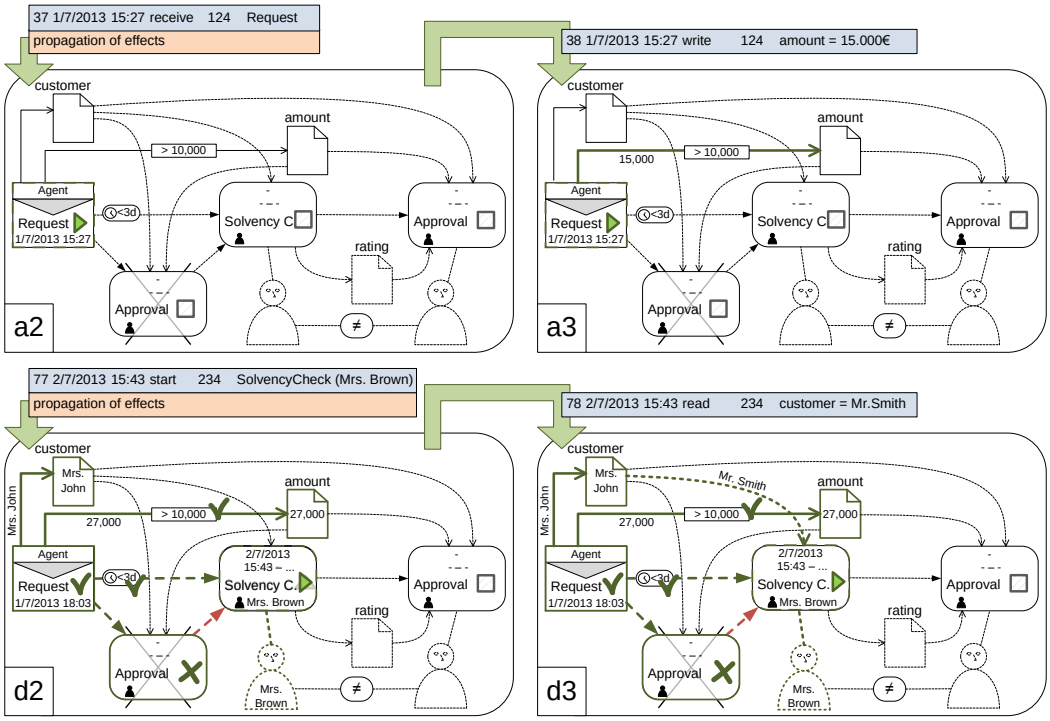


Fig. 13: Handling of data events

5.4.5 Propagation of effects

After each update or event handling, we propagate the corresponding effects (i.e., changes to annotations) on adjacent nodes and edges in order to ensure correct annotations (e.g., activation of subsequent task nodes). Further, we detect contradictory annotations related to the data and resource perspectives (cf. Def. 10).

In particular, resources are propagated from task nodes to Not_Marked (□), dependent resource nodes (r2) by following resource edges (r1). In turn, data values and the identifiers of data containers are propagated from data flow edges to Not_Marked (□), dependent data object nodes (d1) as well as data container nodes (d2). Next, the edge and its target node are marked with Satisfied (✓) (x3).

Note that the aforementioned propagation steps are not performed, if resources, data values or identifiers of target resource nodes, data objects or data containers were already set to another value before. To highlight such conflicts, the corresponding data flow and resource edges will be marked with Skipped (×) (x4). Not_Marked (□) data flow edges will be also marked with Skipped (×) if the corresponding task or message node is completed (d5). Afterwards, all conditions (x6) and relations are reevaluated (x7). If any element of the eCRG, affiliated to a dependent task or message node, is now Skipped (×), the corresponding task or message node will be marked with Skipped (×) as well (x8).

Next, the outgoing sequence flows of completed nodes are marked with Satisfied (✓) (cf1). In turn, Not_Marked (□), incoming sequence flow edges of already started nodes are marked with Skipped (×). Further, sequence flow edges from and to Skipped (×) nodes

are marked with SKIPPED (×) as well (cf2). Task and message nodes become ACTIVATED (△) when all incoming sequence flows, on which it depends, are marked as SATISFIED (✓) satisfied (cf3). In turn, task or message nodes will be marked with SKIPPED (×) when they depend on incoming sequence flows being SKIPPED (×) before (cf4). Note that the latter might trigger (cf2) again.

Definition 10 (Effect Propagation).
Let $M := (m, res, val, id, t_s, t_e) \in \mathfrak{M}_\Psi$ *be a marking of eCRG* Ψ*. Then:* $prop : \mathfrak{M}_\Psi \to \mathfrak{M}_\Psi$, $prop(M) := (m^{(\star)}, res^{(1)}, val^{(1)}, id^{(1)}, t_s, t_e)$ **propagates effects** *on* M*, whereby*

$$res^{(1)}(\lambda) := \begin{cases} \rho, \textit{ if } \left(\lambda = (t,r) \in \overrightarrow{pfm} \wedge m(\lambda) = \square \wedge res(t) = \rho\right) & (r1) \\ \quad \vee \left(\lambda \in R \wedge res(\lambda) = \epsilon \wedge (\exists e := (t,\lambda) \in \overrightarrow{pfm} \wedge \lambda \triangleq e \wedge res(e) = \rho)\right) & (r2) \\ res(\lambda), \textit{ else,} \end{cases}$$

$$val^{(1)}(\lambda) := \begin{cases} v, \textit{ if } \lambda \in O \wedge val(\lambda) = \epsilon \wedge (\exists e := (n,\lambda) \in \overrightarrow{df} : \lambda \triangleq e \wedge res(e) = v) & (d1) \\ val(\lambda), \textit{ else,} \end{cases}$$

$$id^{(1)}(\lambda) := \begin{cases} \omega, \textit{ if } \lambda \in C \wedge id(\lambda) = \epsilon \wedge (\exists e := (n,\lambda) \in \overrightarrow{df} : \lambda \triangleq e \wedge id(e) = \omega) & (d2) \\ id(\lambda), \textit{ else,} \end{cases}$$

$$m^{(1)}(\lambda) := \begin{cases} \checkmark, \textit{ if } \left(\lambda \in R \wedge m(\lambda) = \square \wedge res^{(1)}(\lambda) \neq \epsilon\right) & (x3) \\ \quad \vee \left(\lambda \in O \wedge m(\lambda) = \square \wedge val^{(1)}(\lambda) \neq \epsilon\right) & (x3) \\ \quad \vee \left(\lambda \in C \wedge m(\lambda) = \square \wedge id^{(1)}(\lambda) \neq \epsilon\right) & (x3) \\ \quad \vee \left(\lambda = (t,r) \in \overrightarrow{pfm} \wedge m(\lambda) = \square \wedge res^{(1)}(r) = res^{(1)}(\lambda)\right) & (x3) \\ \quad \vee \left(\lambda = (n,o) \in \overrightarrow{df} \wedge o \in O \wedge m(\lambda) = \square \wedge val^{(1)}(o) = val^{(1)}(\lambda)\right) & (x3) \\ \quad \vee \left(\lambda = (n,c) \in \overrightarrow{df} \wedge c \in C \wedge m(\lambda) = \square \wedge id^{(1)}(c) = id^{(1)}(\lambda)\right) & (x3) \\ \times, \textit{ if } \left(\lambda = (t,r) \in \overrightarrow{pfm} \wedge m(\lambda) = \square \wedge res^{(1)}(r) \neq res^{(1)}(\lambda)\right) & (x4) \\ \quad \vee \left(\lambda = (n,o) \in \overrightarrow{df} \wedge o \in O \wedge m(\lambda) = \square \wedge val^{(1)}(o) \neq val^{(1)}(\lambda)\right) & (x4) \\ \quad \vee \left(\lambda = (n,c) \in \overrightarrow{df} \wedge c \in C \wedge m(\lambda) = \square \wedge id^{(1)}(c) \neq id^{(1)}(\lambda)\right) & (x4) \\ \quad \vee \left(\lambda = (n,o) \in \overrightarrow{df} \wedge o \in O \wedge m(\lambda) = \square \wedge m(n) = \checkmark\right) & (d5) \\ \quad \vee \left(\lambda = (n,c) \in \overrightarrow{df} \wedge c \in C \wedge m(\lambda) = \square \wedge m(n) = \checkmark\right) & (d5) \\ m'(\lambda), \textit{ else,} \end{cases}$$

$$m^{(2)}(\lambda) := \begin{cases} \checkmark, \textit{ if } \left(\lambda \in \diamond \wedge m^{(1)}(at(\lambda)) = \checkmark \wedge \lambda(at(\lambda))\right) & (x6) \\ \quad \vee \left(\lambda = (r_1,r_2) \in \overrightarrow{rr} \cup \wedge m^{(1)}(r_1) = \checkmark \wedge m^{(1)}(r_2) = \checkmark \wedge \lambda(r_1,r_2)\right) & (x7) \\ \quad \vee \left(\lambda = (o_1,o_2) \in \overrightarrow{dr} \cup \wedge m^{(1)}(o_1) = \checkmark \wedge m^{(1)}(o_2) = \checkmark \wedge \lambda(o_1,o_2)\right) & (x7) \\ \times, \textit{ if } \left(\lambda \in \diamond \wedge m^{(1)}(at(\lambda)) = \checkmark \wedge \lambda(at(\lambda))\right) & (x6) \\ \quad \vee \left(\lambda = (r_1,r_2) \in \overrightarrow{rr} \cup \wedge m^{(1)}(r_1) = \checkmark \wedge m^{(1)}(r_2) = \checkmark \wedge \lambda(r_1,r_2)\right) & (x7) \\ \quad \vee \left(\lambda = (o_1,o_2) \in \overrightarrow{dr} \cup \wedge m^{(1)}(o_1) = \checkmark \wedge m^{(1)}(o_2) = \checkmark \wedge \lambda(o_1,o_2)\right) & (x7) \\ m^{(1)}(\lambda), \textit{ else,} \end{cases}$$

$$m^{(3)}(\lambda) := \begin{cases} \times, \textit{ if } \exists \lambda' \in \Lambda : \phi(\lambda') = \lambda \wedge \lambda' \triangleq \lambda \wedge m^{(2)}(\lambda') = \times & (x8) \\ m^{(2)}(\lambda), \textit{ else,} \end{cases}$$

$$m^{(4)}(\lambda) := \begin{cases} \checkmark, \textit{ if } \lambda = (n_1,n_2) \in \overrightarrow{sf} \wedge m(\lambda) = \square \wedge m^{(3)}(n_1) = \checkmark & (cf1) \\ m^{(3)}(\lambda), \textit{ else,} \end{cases}$$

$$m^{(5)}(\lambda) := \begin{cases} \triangle, \textit{ if } \lambda \in T \cup MN \wedge m^{(4)}(\lambda) = \square \\ \qquad \wedge (\forall e = (n,\lambda) \in \overrightarrow{sf} : e \triangleq \lambda \Rightarrow m^{(4)}(e) = \checkmark) & (cf3) \\ m^{(4)}(\lambda), \textit{ else,} \end{cases}$$

$$\text{for } i \geq 6\text{:}\quad m^{(i+1)}(\lambda) := \begin{cases} \times, \text{ if } \left(\lambda = (n_1, n_2) \in \overrightarrow{sf} \wedge m^{(i)}(\lambda) = \square \wedge m^{(i)}(n_1) = \times\right) & (cf2) \\ \quad \vee \left(\lambda = (n_1, n_2) \in \overrightarrow{sf} \wedge m^{(i)}(\lambda) = \square \wedge m^{(i)}(n_2) \in \{\blacktriangleright, \checkmark, \times\}\right) & (cf2) \\ \quad \vee \left(\lambda \in T \cup MN \wedge m^{(i)}(\lambda) = \square\right. & \\ \qquad \left. \wedge\, (\exists e = (n, \lambda) \in \overrightarrow{sf} : e \triangleq \lambda \wedge m^{(i)}(e) = \times)\right) & (cf4) \\ m^{(i)}(\lambda), \text{ else, and} \end{cases}$$

$m^{(\star)} := m^{(j)}(j := min\{i \in \mathbb{N} | i \geq 6 \wedge m^{(i)} = m^{(i+1)}\})$ *is the final marking, which is reached, when the propagation terminates.*

Figs. 14-17 highlight the propagation after updates as well as after the handling of start, message, end, and data events. In particular, Fig. 14 transfers the annotation SKIPPED (×) of the time condition on the corresponding sequence flow edge that was marked with SATISFIED (✓) before. Based on this change, the subsequent task node *solvency check* is marked with SKIPPED (×) as well and its NON_MARKED (□), incoming and outgoing sequence flow edges. Finally, the annotation SKIPPED (×) is propagated to the consequence absence task node *approval* as well as the consequence occurrence task node *approval*. Hence, the final marking Fig. 14(B'') shows a temporal conflict.

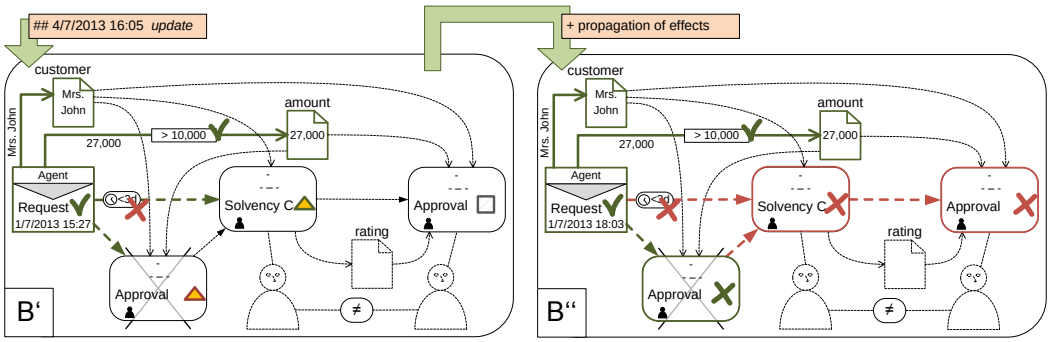


Fig. 14: Effect propagation after updates and temporal conflict

In Fig. 15, the effects of a start event are propagated. In particular, the adjacent resource edge and resource node become COMPLETED (✓) and are annotated with resource *Mrs. Brown*. Further, the NON_MARKED (□), incoming sequence flow edge of the just started task node *solvency check* is marked with SKIPPED (×). Finally, the consequence absence task node *approval* becomes also SKIPPED (×).

Fig. 16 propagates the effects of an end event. In particular, the outgoing sequence flow edges of the just completed message node *request* are marked with SATISFIED (✓). Furthermore, the subsequent task nodes *solvency check* and *approval* become ACTIVATED (△).

The effects of data events are propagated in Fig. 17. In the upper example, the value *15,000* of the data flow edge used is written on the target data object *amount* that also becomes SATISFIED (✓). Further, the antecedence data condition the respective data flow is successfully evaluated and hence also marked as satisfied. As opposed to this, the comparison of the data object *customer* and its just written outgoing data flow edge fails. Hence, the latter data flow edge changes from SATISFIED (✓) to SKIPPED (×). Due to this change, the subsequent task node *solvency check* is marked with SKIPPED (×) as well and its outgoing

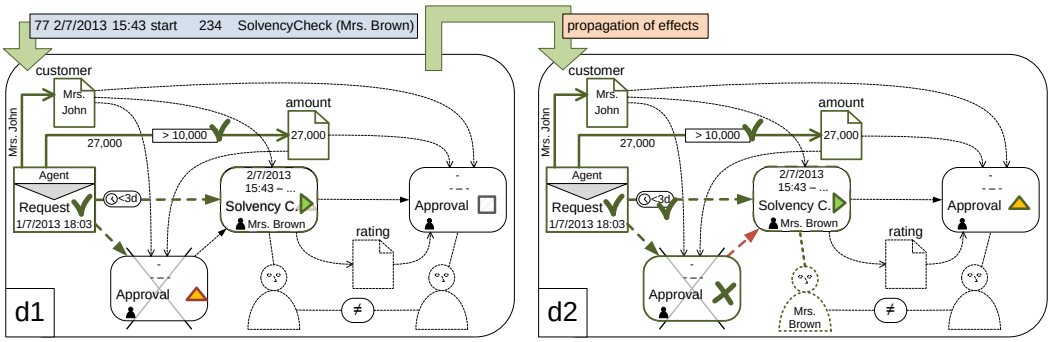


Fig. 15: Effect propagation after handling start events

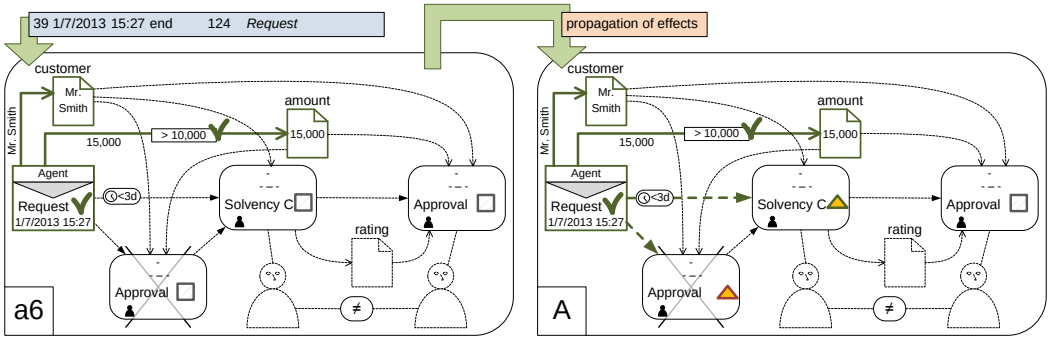


Fig. 16: Effect propagation after handling end events

sequence flow edge. Finally, the annotation SKIPPED (×) is also propagated on the following consequence occurrence task node *approval*. Hence, Fig. 14(D) highlights a data conflict.

Finally, Def. 11 formally specifies the update of the state of an eCRG to the current point in time as well as the processing of a particular event based on Defs. 5-10 and Fig. 9.

Definition 11 (Event processing procedure).
Let $\mathcal{M}_\Psi^{\mathcal{L},i}$ be the state of an eCRG Ψ after processing event stream $\mathcal{L}$ until the ith position ($i \in \mathbb{N}$), and let $\sigma := \mathcal{L}(i+1)$ be the upcoming event on the following position of $\mathcal{L}$ and $\vartheta := time(\sigma) \in \mathfrak{T}$ be the current point in time. Then:

- $\mathcal{M}_\Psi^{\mathcal{L},i\to\vartheta} := \{M' | M' = prop(upd^\vartheta(M)) \wedge M \in \mathcal{M}_\Psi^{\mathcal{L},i}\}$ *is the state of Ψ after processing event stream $\mathcal{L}$ until position i and after the update to the current point in time ϑ and*
- $\mathcal{M}_\Psi^{\mathcal{L},i+1} := \{prop(M') | M' \in hdl^\sigma(M) \wedge M \in \mathcal{M}_\Psi^{\mathcal{L},i\to\vartheta}\}$ *is the state of Ψ after processing σ, i.e. after processing event stream $\mathcal{L}$ until position $i+1$.*

Table 3 outlines the set of markings that results when completely processing the event stream from Fig. 1 for compliance rule c_1. Note that marking F ensures that c_1 is satisfied for the request of Mr. Smith as highlighted in Fig. 18. In turn, Figs. 14(B''), 17(D), 18(I), and 18(J) highlight conflicts regarding the data, control flow, time, and resource perspectives.

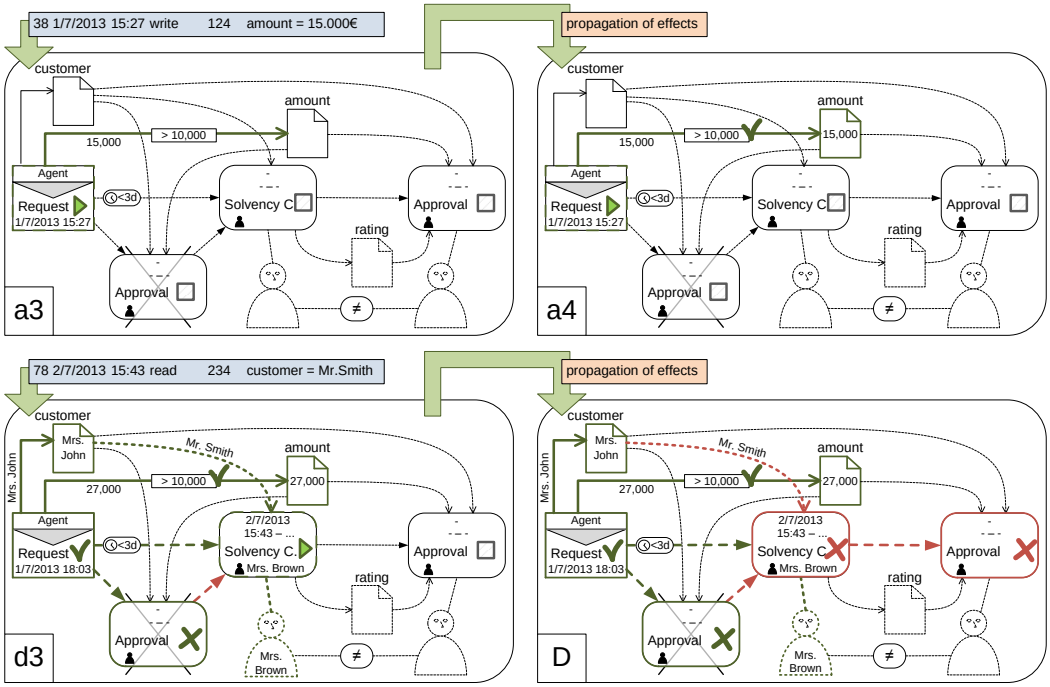


Fig. 17: Effect propagation after handling data events and data conflict

Note that such conflicts only indicate why the considered events do not constitute a solution of a particular compliance rule. However, there might be another set of events that provide a satisfaction. In turn, Fig. 15($d2$) indicates, which data values shall be read by task *solvency check* and how task *approval* shall be performed afterwards in order to satisfy c_1. Hence, Fig. 15($d2$) may be utilized as recommendation to users in order to proactively ensure compliance.

Table 3: Compliance state $\mathcal{M}_{c_1}^{\mathcal{L},99}$

#	Request		Approval (CA)			Solvency Check			Approval (CO)			customer	cust→ App.(CA)	cust→ Solv.C	amount	rating	Activated
0	△		□			□			□			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	✓	129	✗	642	Brown	△			□			Smith	John	ϵ	25.000	ϵ	

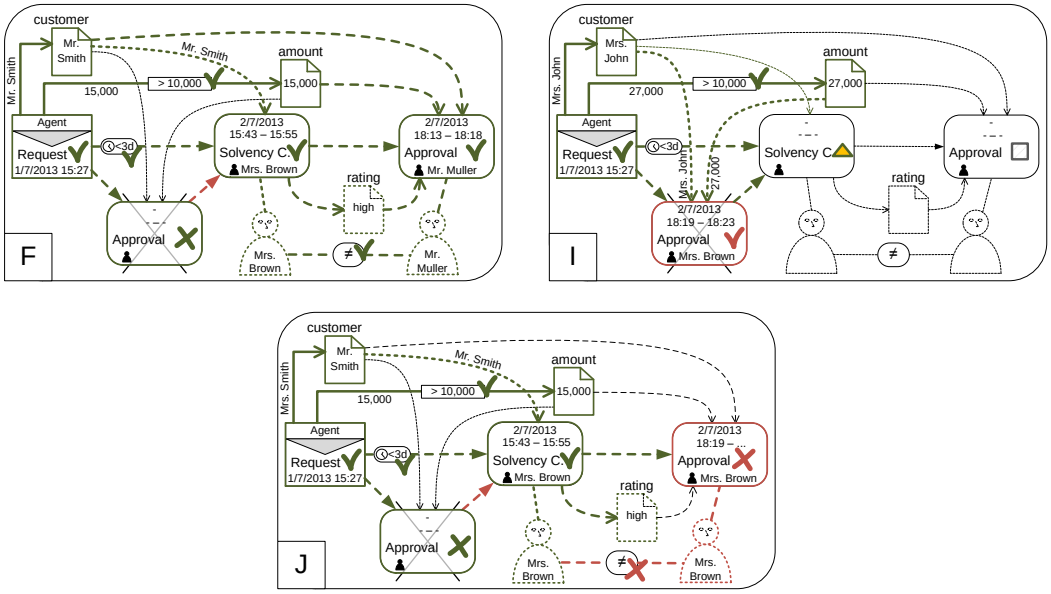


Fig. 18: Fulfilling marking as well as control-flow and resource conflicts

6 Compliance Analytics

Another fundamental requirement of any compliance monitoring approach is to provide detailed user feedback. In this context, Section 6.1 introduces well-defined compliance assessments that not only highlight compliance violations, but also identify their causes. In addition, Section 6.2 shows how compliance metrics can be specified and how corresponding measurements can be obtained. Finally, Section 6.3 provide recommendations that support users in selecting the next process steps, whose execution will ensure compliance.

6.1 Compliance Assessments

Providing compliance assessments requires the analysis of the state of an eCRG and the markings it contains in order to clearly determine the corresponding states of compliance; i.e., either NOT_ACTIVATED, VIOLABLE, SATISFIED, PENDING, or VIOLATED (cf. Section 5.2). For this purpose, first of all, Def. 12 introduces a partial order over markings. The latter allows clustering the markings that correspond to each other as well as to specify and distinguish between different activations of an eCRG in Def. 13. Based on the latter, Def.14 finally provides compliance assessments; i.e., calculates the state of compliance (cf. Section 5.2) in order to enable simple and intuitive classifications.

Definition 12 (Extensions of Markings).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_\Psi$ the current state of compliance of Ψ. Further, let $M = (m, res, val, id, t_s, t_e)$ and $M' = (m', res', val', id', t'_s, t'_e)$ be two markings of Ψ; i.e., $M, M' \in \mathcal{M}$. Then: We say

- *We say M is* **extended** *by M' and write $M \preceq M'$, iff M' only differs from M in markings that are neither* RUNNING *($\blacktriangleright$) nor* COMPLETED *($\checkmark$) in M, i.e.,*

$$M \preceq M' :\Leftrightarrow \forall n \in T \cup M : m(n) \notin \{\blacktriangleright, \checkmark\} \vee id(n) = id'(n)$$

Definition 13 (Activated Markings).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_\Psi$ be the current state of compliance of Ψ. Further, let $M = (m, res, val, id, t_s, t_e)$ be a marking of Ψ; i.e., $M, M' \in \mathcal{M}$. Then: We say

- *M is* **activated** *or an* **activation** *of Ψ (written as* ACTIVATED(M)*), iff M marks each element of the antecedence occurrence pattern as satisfied, but does not satisfy any element of the antecedence absence pattern. Furthermore, neither extends M another activated marking nor exists a marking in $\mathcal{M}$ that extends M and satisfies a condition of the antecedence absence pattern.*

$$\begin{aligned}\text{ACTIVATED}(M) :\Leftrightarrow \big(& (\forall \lambda \in \Lambda_\Psi^{AO} : m(\lambda) = \checkmark) \\ & \wedge (\forall \alpha \in \Gamma_\Psi^{AA,CO,CA} : m(\alpha) \in \{\square, \triangle, \times\}) \wedge id(\alpha) = \epsilon \big) \\ & \wedge \big(\forall\, M_2 \in \mathcal{M} \text{ with } M \preceq M_2 : \forall \alpha \in \Gamma_\Psi^{AA} : m_2(\alpha) \neq \checkmark \big) \\ & \wedge \big(\forall\, M_3 \in \mathcal{M} \text{ with } M_3 \preceq M : \neg \text{ACTIVATED}(M_3) \big) \big)\end{aligned}$$

Definition 14 (Compliance Assessments).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_\Psi$ be the current state of Ψ. Further, let $M \in \mathcal{M}$ be an activated marking; i.e. Activated(M).
Then: We say

- *M is* **temporally satisfied** *(written as* TempSatisfied(M)*), iff there exists a marking M_2 that extends M and satisfies the consequence; i.e. M_2 marks each element of the consequence occurrence pattern with* Satisfied$(\checkmark)$ *and does not satisfy any element of the consequence absence pattern. Further, there exists no marking in $\mathcal{M}$ that extends M_2 and satisfies a condition of the consequence absence pattern.*

$$\begin{aligned}\textsc{TempSatisfied}(M) :\Leftrightarrow \big(\ & \exists\, M_2 \in \mathcal{M} \text{ with } M \preceq M_2 : (\forall \lambda \in \Lambda_\Psi^{CO} : m_2(\lambda) = \checkmark) \\ & \wedge (\forall \alpha \in \Gamma_\Psi^{CA} : m_2(\alpha) \in \{\square, \triangle, \times\} \wedge id(\alpha) = \epsilon) \\ & \wedge \big(\ \forall\, M_3 \in \mathcal{M} \text{ with } M \preceq M_2 \preceq M_3 : \forall \alpha \in \Gamma_\Psi^{CA} : m_3(\alpha) \neq \checkmark\big)\big)\end{aligned}$$

- *M is* **violable (violably satisfied)** *(written as* Violable(M)*), iff M is temporally satisfied, but for each M_2 extending M and satisfying the consequence pattern, there remains at least one consequence absence node that has not been marked with* Skipped *($\times$) yet; i.e. that node might be executed in the following steps.*

$$\begin{aligned}\textsc{Violable} :\Leftrightarrow\ & \textsc{TempSatisfied}(M) \\ & \wedge \big(\ \forall\, M_2 \in \mathcal{M} \text{ with } M \preceq M_2 : (\forall \lambda \in \Lambda_\Psi^{CO} : m_2(\lambda) = \checkmark) \\ & \quad \wedge\ (\forall \alpha \in \Gamma_\Psi^{CA} : m_2(\alpha) \in \{\square, \triangle, \times\} \wedge id(\alpha) = \epsilon) \\ & \quad \wedge \big(\ \forall\, M_3 \in \mathcal{M} \text{ with } M \preceq M_2 \preceq M_3 : \forall \alpha \in \Gamma_\Psi^{CA} : m_3(\alpha) \neq \checkmark\big) \\ & \quad \wedge\ (\exists \alpha \in \Gamma_\Psi^{CA} : m_2(\alpha) \in \{\square, \triangle\})\big)\end{aligned}$$

- *M is* **(permanently) satisfied** *(written as* Satisfied(M)*), iff M is temporally satisfied, but not violable; i.e.,*

$$\textsc{Satisfied}(M) :\Leftrightarrow \textsc{TempSatisfied}(M) \wedge \neg \textsc{Violable}(M)$$

- *M is* **(permanently) violated** *(written as* Violated(M)*), iff each marking M_2 extending M marks an element of the consequence occurrence pattern with* Skipped *($\times$) or there exists a marking M_3 extending M_2 and satisfying a condition of the consequence absence pattern.*

$$\begin{aligned}\textsc{Violated}(M) :\Leftrightarrow \big(\ & \forall\, M_2 \in \mathcal{M} \text{ with } M \preceq M_2 : (\exists \lambda \in \Lambda_\Psi^{CO} : m_2(\lambda) = \times) \\ & \vee \big(\ \exists\, M_3 \in \mathcal{M} \text{ with } M \preceq M_2 \preceq M_3 : \exists \alpha \in \Gamma_\Psi^{CA} : m_3(\alpha) = \checkmark\big)\big)\end{aligned}$$

- *M is* **pending** *(written as* Pending(M)*), iff M is neither (temporally) satisfied nor violated yet.*

$$\textsc{Pending}(M) :\Leftrightarrow \neg \textsc{TempSatisfied}(M) \wedge \neg \textsc{Violated}(M)$$

Table. 4 applies Def. 14 to markings A and B from Table 3. Note that A and B are the only activations of c_1 in $\mathcal{M}_{c_1}^{\mathcal{L},99}$, since all other markings extend either marking A or B (except the initial marking 0). In particular, Table 4 shows that c_1 is activated twice; once satisfied and once violated. Further, Table 4 emphasises the events that complete the activations (39+58), the fulfillment (95), and the violation (99).

Table 4: Compliance assessments and metrics

#	Extensions	Activated	TempSatisfied	Violable	Satisfied	Violated	Pending
A	{A, C, F, G}	39-...	95-...		95-...		39-95
B	{B, D, E, H}	58-...				99-...	58-99

6.2 Compliance Metrics and Measures

In the context of multiple instances and activations of compliance rules, providing compliance assessments for single activations is no longer sufficient. In turn, additional summaries on an higher level are required. For this purpose, compliance assessments (cf. Sec. 6.1) are combined to realize more sophisticated compliance metrics and measures. First of all, this section formally specifies the cardinality of properties in Def. 15 and, then, shows how the latter can be utilized for calculating compliance metrics and measures.

Definition 15 (Cardinality of a Property).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_\Psi$ be the current state of compliance of compliance rule Ψ. Then:

$\#_\Psi^{\text{Prop}} := |\{M \in \mathcal{M}_\Psi | \text{Prop}(M)\}|$ *is the* **cardinality** *of property* Prop*; i.e., the number of markings with property* Prop $\in$ {Activated, TempSatisfied, Violable, Satisfied, Violated, Pending}.

Note that it is easy to specify metrics based on Def. 15. As example consider Table. 5, which refers to 3 metrics. In particular, the *compliance rate* μ_1, the *critical rate* μ_2, and the *violation rate* μ_3 are defined and calculated with respect to the motivating example from Section 3. (cf. Fig. 2 and Table 4).

Table 5: Compliance assessments and metrics

			compliance rate $\mu_1(c_1)$	critical rate $\mu_2(c_1)$	violation rate $\mu_3(c_1)$
#	date	time	$\mu_1(c_1) = \frac{\#_{c_1}^{\text{TempSatisfied}}}{\#_{c_1}^{\text{Activated}}}$	$\mu_2(c_1) = \frac{\#_{c_1}^{\text{Violable}} + \#_{c_1}^{\text{Pending}}}{\#_{c_1}^{\text{Activated}}}$	$\mu_3(c_1) = \frac{\#_{c_1}^{\text{Violated}}}{\#_{c_1}^{\text{Activated}}}$
0	1/7/2013	15:00	n.d.	n.d.	n.d.
1	1/7/2013	17:00	$\frac{0}{1} = 0.00$	$\frac{0+1}{1} = 1.00$	$\frac{0}{1} = 0.00$
2	1/7/2013	19:00	$\frac{0}{2} = 0.00$	$\frac{0+2}{2} = 1.00$	$\frac{0}{2} = 0.00$
3	2/7/2013	17:00	$\frac{0}{2} = 0.00$	$\frac{0+2}{2} = 1.00$	$\frac{0}{2} = 0.00$
4	2/7/2013	18:18	$\frac{1}{2} = 0.50$	$\frac{0+1}{2} = 0.50$	$\frac{0}{2} = 0.00$
5	2/7/2013	19:00	$\frac{1}{2} = 0.50$	$\frac{0+0}{2} = 0.00$	$\frac{1}{2} = 0.50$

Table. 5 shows the progress of the compliance rate $\mu_1(c_1)$, critical rate $\mu_2(c_1)$ and violation rate $\mu_3(c_1)$ for compliance rule c_1 over time and along the event log from Fig. 2. As long as no request is received, there is no activation of c_1 and hence all rates are not yet defined (cf. line 0). Due the request from *Mr. Smith* at 1/7/2013 – 15:27 compliance rule c_1

is activated once in state PENDING. Therefore, the critical rate $\mu_2(c_1)$ becomes 1.00, but the compliance rate $\mu_1(c_1)$ as well as violation rate $\mu_3(c_1)$ become 0.00 (cf. line 1). In turn, the values of $\mu_1(c_1)$, $\mu_2(c_1)$ and $\mu_3(c_1)$ are neither changed by the request from *Mrs. John* at 1/7/2013 – 18:03, which activates c_2 the second time, nor by the solvency check at 2/7/2013 – 15:43 (cf. line 2+3). This is due to the fact that the second activation of c_1 starts also in state pending as well as the first activation remains pending even after the solvency check. However, when the request of *Mr. Smith* is approved at 2/7/2013 – 18:13, the corresponding activation is SATISFIED. Hence, the compliance rate $\mu_1(c_1)$ increases to 0.50 and critical rate $\mu_2(c_1)$ decreases to 0.50 as well, whereas the violation rate $\mu_3(c_1)$ stagnates at 0.00 (cf. line 4). Finally, the second activation of c_1 becomes VIOLATED by the approval at 2/7/2013 – 18:19. Accordingly, no activation remains PENDING and the critical rate $\mu_2(c_1)$ drops to 0.00 In turn, the violation rate $\mu_3(c_1)$ increases to 0.50 as the compliance rate $\mu_1(c_1)$ did before.

6.3 Recommendations

Section 5.4 argues that steps ensuring the compliant continuation of business process instances can be recommend based on markings. However, this requires the selection of appropriate markings. Note that the simple approach taking the marking that provides the most extensive extension of an activation does not always fit. Consider Table 6, which provides the set of markings $\mathcal{M}_{\Psi}^{95}$ after processing the event log from Fig. 2 up to event 95. For both activations A and B there is an extending marking ($A \preceq C$ and $B \preceq D$). However, in the first case, C provides the adequate recommendation, whereas in the second case the recommendation is not D, but B.

Table 6: Compliance state $\mathcal{M}_{c_1}^{\mathcal{L},80}$

#	Request		Approval (CA)		Solvency Check			Approval (CO)		customer	cust→ App.(CA)	cust→ Solv.C	amount	rating	ACTIVATED
0	△		□		□			□		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	

Def. 16 provides a more advanced and proper selection of markings that may then be used as recommendation.

Definition 16 (Recommendations).
Let $\Psi = (N, E, \diamond, type, src, tgt, at, pt)$ be an eCRG and $\mathcal{M}_{\Psi}$ be the current state of compliance rule Ψ. Further, let $M \in \mathcal{M}_{\Psi}$ be an activated and not violated marking; i.e., it holds: $\text{ACTIVATED}(M) \wedge \neg\text{VIOLATED}(M)$. *Then: We say*

- *$M_2 \in \mathcal{M}_{\Psi}$ is a* **simple recommendation** *for activation M of compliance rule Ψ (written as $M \preceq_r M_2$), iff M_2 extends M and M_2 neither skips any part of the consequence occurrence pattern nor satisfies any part of the consequence absence pattern nor it there is another marking $M_3 \in \mathcal{M}_{\Psi}$ further extending M_2, i.e.,*

$$\begin{aligned} M \preceq_r M_2 :\Leftrightarrow\ & M \preceq M_2 \\ & \wedge \left(\nexists \alpha \in \Gamma_{\Psi}^{CO} : m_2(\alpha) = \times\right) \\ & \wedge \left(\nexists M_3 \in \mathcal{M} \text{ with } M \preceq M_2 \preceq M_3 : \exists \alpha \in \Gamma_{\Psi}^{CA} : m_3(\phi(\alpha)) = \checkmark\right) \end{aligned}$$

- $M_2 \in \mathcal{M}_\Psi$ *is a* **(real) recommendation** *for the activation of* Ψ *through marking* M *(written as*$M \preceq_\mathfrak{R} M_2$*), iff* M_2 *is a simple recommendation for* M *and there is no other recommendation* $M_3 \in \mathcal{M}_\Psi$ *for* M *extending* M_2*, i.e.,*

$$M \preceq_\mathfrak{R} M_2 :\Leftrightarrow M \preceq_r M_2 \\ \wedge \nexists M_3 \in \mathcal{M} \text{ with } M \preceq_r M_3 : M_2 \prec M_3 \text{ (i.e., } M_2 \preceq M_3 \wedge M_2 \neq M_3\text{)}$$

- $\mathfrak{R} : \mathcal{M}_\psi \to 2^{\mathcal{M}_\psi} : \mathfrak{R}(M) := \{M' \in \mathcal{M}_\Psi | M \preceq_\mathfrak{R} M'\}$ *is the set of* **all recommendations** *for* M*.*

Note that each marking M*, which is* Activated *and not* Violated*, must have at least one simple and, hence, also one real recommendation. Otherwise, it would match the definition of* Violated *(cf. Def. 14).*

The application of Def. 16 on our example results in $\mathfrak{R}(A) = \{C\}$ and $\mathfrak{R}(B) = \{B\}$. Note that the application of $\mathfrak{R}$ to an activation might still result in more than one recommendation. In this case, each recommendation corresponds to another, already started solution approach ensuring compliance with the respective rule. However, if only a single recommendation is desired, the most advanced recommendation (i.e., satisfying most consequence occurrence nodes) or the latest one can easily be selected.

7 Summary and Outlook

This report introduces an operational semantics for the extended compliance rule graph (eCRG) language [17, 18]. Beyond the control flow perspective, the latter supports the monitoring of the data, time, and resource perspectives as well as the monitoring of the interactions with partners. In particular, eCRGs are annotated with text, colors and symbols in order to visually highlight the current state of compliance. Furthermore, formal definitions specify how observed events continuously change and evolve these annotations. Furthermore, formal criteria for assessing compliance are provided, which, in turn, constitute the basis for compliance metrics introduced as examples. Finally, the recommendations are provided, which support users in selecting the next process steps, whose execution will ensure compliance.

As opposed to other approaches enabling the monitoring of business process compliance at run-time, the operational semantics of the eCRG language supports all 10 compliance monitoring functionalities (CMF) that have been proposed in [5] (cf. Table 7). In particular, full support of the control flow, data, time, and resource perspectives (CMF 1-3) is provided as well as interactions with partners are considered. The proposed approach assumes activities to be non-atomic, but stateful (CMF 4+5). Different instantiations (i.e., activations) of compliance rules are explicitly identified and are further extended to highlight the causes of compliance violations (CMF 6+9). Beyond detecting the latter (i.e. reactive monitoring), recommendations (i.e., proactive monitoring) are provided as well (CMF 7+8). Finally, the operational semantics of the eCRG language builds a suitable basis for the specification of compliance metrics in order to measure different degrees of compliance (CMF 10).

Table 7: Compliance monitoring functionalities [5]

Approach	CMF 1 time	CMF 2 data	CMF 3 resources	CMF 4 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 [16]	+/-	-	-	+	-	-	+	+	+	+/-
Mubicon EC [15]	+	+/-	+	+	+	+	+	-	+/-	+/-
ECE Rules [26]	+	+/-	+	+	-	-	+	-	+/-	+
SCT [24]	+/-	-	+	+	+	-	-	+	-	-
SeaFlows [10]	+/-	+/-	+/-	+	+/-	+	+	+	+	+/-
eCRG Op. Semantics	+	+	+	+	+	+	+	+	+	+

Next steps will be the implementation of a proof-of-concept prototype in order to evaluate the operational semantics for the eCRG language.

Note that this work was done within the research project C^{3}Pro that deals with change and compliance in cross-organizational business processes [21]. Accordingly, our overall aim is to ensure multi-perspective compliance for all phases of the process life cycle. Hence, we will investigate *a priori* compliance checking with the eCRG at design time as well as we plan to consider compliance checking in the context of cross-organizational process changes and change propagation [44].

References

1. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
2. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN'97. Volume 1248 of LNCS., Springer (1997) 407–426
3. Reichert, M., Dadam, P.: A framework for dynamic changes in workflow management systems. In: DEXA'97. (1997) 42–48
4. Ly, L.T., Knuplesch, D., Rinderle-Ma, S., Göser, K., Pfeifer, H., Reichert, M., Dadam, P.: SeaFlows toolset - compliance verification made easy for process-aware information systems. In: Information Systems Evolution - CAiSE Forum 2010. Volume 72 of LNBIP., Springer (2011) 76–91
5. 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
6. Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: BPM'07. Volume 4714 of LNCS., Springer (2007) 149–164
7. Lenz, R., Reichert, M.: IT support for healthcare processes premises, challenges, perspectives. Data & Knowledge Engineering **61**(1) (2007) 39–58
8. Knuplesch, D., Reichert, M.: Ensuring business process compliance along the process life cycle. Technical Report 2011-06, Ulm University (2011)
9. 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
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. (2011) 82–99
11. Namiri, K., Stojanovic, N.: Pattern-Based design and validation of business process compliance. In: CAiSE'07. Volume 4803 of LNCS., Springer (2007) 59–76
12. Alles, M., Kogan, A., Vasarhelyi, M.: Putting continuous auditing theory into practice: Lessons from two pilot implementations. Information Systems **22**(2) (2008) 195–214
13. 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
14. Ramezani Taghiabadi, E., Fahland, D., van Dongen, B.F., van der Aalst, W.M.P.: Diagnostic information for compliance checking of temporal compliance requirements. In: CAiSE'13. Volume 7908 of LNCS., Springer (2013) 304–320
15. Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P.: Monitoring business constraints with the event calculus. Transactions on Intelligent Systems and Technology **5**(1) (2014) 17.1–17.30
16. 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. (2011) 132–147
17. 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
18. 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
19. Kharbili, M.E., de Medeiros, A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: MobIS'08. (2008) 107–113
20. Becker, J., Delfmann, P., Eggert, M., Schwittay, S.: Generalizability and applicability of model-based business process compliance-checking approaches. BuR - Business Research **5**(2) (2012) 221–247
21. 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
22. Accorsi, R.: An approach to data-driven detective internal controls for processaware information systems. In: DUMW'12. (2012) 29–33

23. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-based monitoring of process execution violations. In: BPM'11. Volume 6896 of LNCS., Springer (2011) 182–198
24. Santos, E., 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
25. 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
26. 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
27. Liu, R., Vaculn, R., Shan, Z., Nigam, A., Wu, F.: Business artifact-centric modeling for real-time performance monitoring. In: BPM'11 Workshops. Volume 6896 of LNCS., Springer (2012) 265–280
28. 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
29. Outmazgin, N., Soffer, P.: A process mining-based analysis of business process work-arounds. Software & Systems Modeling (2014) 1–15
30. 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
31. 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
32. 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
33. 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
34. Accorsi, R., Lowis, L., Sato, Y.: Automated certification for compliant cloud-based business processes. Business & Information Systems Engineering **3**(3) (2011) 145–154
35. Kumar, A., Yao, W., Chu, C.: Flexible process compliance with semantic constraints using mixed-integer programming. INFORMS Journal on Computing **25**(3) (2013) 543–559
36. Ly, L.T., Rinderle-Ma, S., Göser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. Information Systems Frontiers **14**(2) (2012) 195–219
37. Ramezani, E., Fahland, D., van der Werf, J.M., Mattheis, P.: Separating compliance management and business process management. In: BPM'11 Workshops. Volume 100 of LNBIP., Springer (2012) 459–464
38. Koetter, F., Kochanowski, M., Weisbecker, A., Fehling, C., Leymann, F.: Integrating compliance requirements across business and it. In: EDOC'14, IEEE (2014)
39. Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Hints on how to face business process compliance. Jornadas de Ingeniera del Software y Bases de Datos **4**(4) (2010) 26–32
40. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: CAiSE'10. Volume 6051 of LNCS., Springer (2010) 9–23
41. Ly, L.T.: SeaFlows - A Compliance Checking Framework for Supporting the Process Lifecycle. Dissertation Thesis, University of Ulm, Germany (2013)
42. 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)
43. Gou, Y., Ghose, A.K., Chang, C.F., Dam, H.K., Miller, A.: Semantic monitoring and compensation in socio-technical processes. In: ER'2014 Workshops. Volume 8823 of LNCS., Springer (2014) 117–126
44. Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Information Systems **49** (April 2015) 1–24