Ensuring Business Process Compliance Along the Process Life Cycle

David Knuplesch and Manfred Reichert

Abstract
Business processes are subject to semantic constraints that stem from regulations, laws and guidelines, and are also known as *compliance rules*. Hence, process-aware information systems have to ensure compliance with those rules in order to guarantee semantically correct and error-free executability as well as changeability of their business processes. This report discusses how compliance rules can be defined and how business process compliance can be ensured for the different phases of the process lifecycle.

1 Motivation

In many past publications [1, 2] the correctness of a pre-specified process model was directly related to its syntactical properties and behavioral soundness (i.e., state consistency). However, these are not the only constraints, a pre-specified process model has to obey. Typically, process models and corresponding process instances are also subject to semantic constraints stemming from a variety of sources like standards, regulations, guidelines, corporate policies, and laws (e.g. Basel or Sarbanes-Oxley-Act). In the following these semantic constraints are denoted as *compliance rules*, and techniques for ensuring the compliance of a business process with these rules are covered under the term *business process compliance.*
Compliance rules typically restrict the order in which process activities may be executed. Hence, a compliance rule can be defined as a function that recognizes whether or not a process instance – represented by its execution trace – complies with the rule (cf. Definition 1). Generally, syntactically correct and sound process models still can violate compliance rules. When being confronted with large process models or numerous compliance rules, however, traditional approaches like *manual auditing* are not feasible. This, in turn, raises the demand for techniques automatically ensuring business process compliance in all phases of the process lifecycle.

Definition 1 (Compliance Rule). Let Σ be the set of all activities and let $\Sigma^\star$ be the set of all possible execution traces of processes based on activities from Σ. Then: A *compliance rule* ϕ defines a function $\phi : \Sigma^\star \rightarrow Boolean$ that considers any trace $\sigma =< e_1, \ldots, e_k > \in \Sigma^\star$ either to be true (i.e., to be compliant with ϕ) or false (i.e. to violate ϕ or to be not compliant with it). We further denote $\sigma \models \phi :\Leftrightarrow \phi(\sigma)$ and say trace σ satisfies compliance rule ϕ.

Example 1 (Compliance Rules). Consider the process model S_{med} from Figure 1. It shows a pre-specified process model for planning and performing a keyhole surgery in a hospital. Further, consider the informal compliance rules from Table 1, which must be satisfied by all medical processes of the respective hospital. In particular, these compliance rules have to be obeyed by the pre-specified process model from Figure 1

Institute of Databases and Information Systems
Ulm University, Germany
e-mail: `{david.knuplesch,manfred.reichert}@uni-ulm.de`

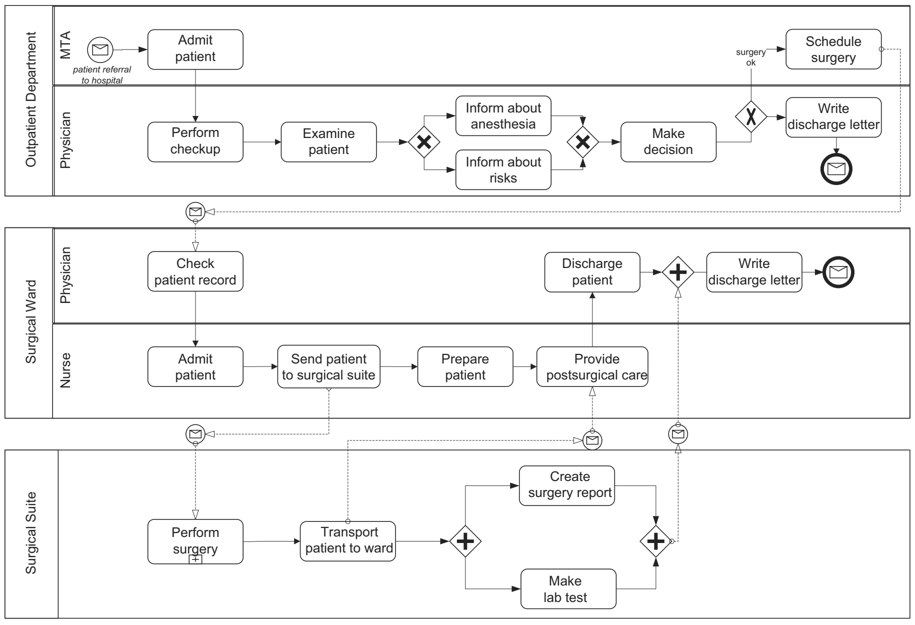


Fig. 1: Pre-Specified Process Model S_{med}

as well. When analyzing the dynamic behavior of the process model, its soundness [1, 2] can be easily verified. However, having a closer look at the model and the compliance rules from Table 1, one can recognize that the process model contains semantic errors; i.e., it violates some of the given compliance rules. For example, according to the process model the *surgical ward* may send the patient to the *surgical suite* before he is prepared; the surgery could be even performed without having prepared the patient at all. Obviously, this violates compliance rule c_1. Further, in the given process model the patient is either informed about anesthesia or risks, but not about both. However, according to compliance rule c_3 the patient must be always informed about the risks after the examination. Hence, c_3 is potentially violated.

Table 1: Examples of Compliance Rules for Medical Processes

c_1	Before a surgery may be performed the patient has to be prepared for it and be sent to the *surgical suite*.
c_2	After examining the patient a decision has to be made. However, this must not be done before the examination.
c_3	After the examination, the patient has to be informed about the risks of the (planned) surgery.
c_4	Before scheduling the surgery the patient has to be informed about anesthesia.
c_5	If a surgery has not been scheduled it must not be performed.
c_6	After a patient is discharged a discharge letter has to be written.
c_7	After performing the surgery and before writing the discharge letter, a surgery report must be created and a lab test be made.

Generally, ensuring business process compliance not only concerns the modeling phase of the process lifecycle [3], i.e., the definition of pre-specified process models. Additionally, compliance has to be monitored for process instances during their execution. This is crucial for process instances being defined or adapted on-the-fly [4], i.e., for which there is no fully pre-specified process model. Further, compliance monitoring at run-time is required if *a priori compliance checking* is not feasible, e.g., if the process model is too large or the compliance rules are too complex. Finally, for completed process instances, a PAIS needs to be able to determine whether or not these instances were executed in compliance with given regulations, laws and guidelines. For this purpose, execution logs need to be analyzed accordingly.

Independent from the process lifecycle phase for which business process compliance has to be ensured, compliance rules have to be specified in a machine-readable way. Hence, this report first deals with issues related to the modeling of compliance rules in Section 2. Following this, it will be shown how compliance can be ensured during the different phases of the process lifecycle. More precisely, Section 3 addresses *a priori* compliance checking in the process modeling phase. Then, Section 4 shows how compliance rules can be monitored during the execution of process instances, whereas Section 5 discusses issues related to the compliance of completed process instances. Section 6 further illustrates how compliance can be ensured in the context of process changes. We address the user perspective in Section 7 and present existing approaches enabling business process compliance in Section 8. The report closes with a summary in Section 9.

2 Modeling Compliance Rules

As prerequisite for verifying business process compliance of pre-specified process models, process instances or process execution logs, corresponding compliance rules need to be provided in a machine-readable way. In literature, there exist different approaches for this. One way to define and represent compliance rules is the usage of *Linear Temporal Logic* (LTL) [5]. LTL is a modal temporal logic with modalities referring to time. It enhances ordinary propositional logic with additional temporal operators as specified in Definition 2.

Definition 2 (Syntax of Linear Temporal Logic). A formula $<LTL>$ is a syntactical correct *LTL formula* if it complies with the following grammar (expressed in BNF):

$$\begin{aligned} <LTL> ::= \ & \top \mid \bot \mid \neg <LTL> \mid (<LTL>) \\ & \mid \mathbf{X} <LTL> \mid \mathbf{F} <LTL> \mid \mathbf{G} <LTL> \\ & \mid <LTL> \wedge <LTL> \mid <LTL> \Rightarrow <LTL> \\ & \mid <LTL> \vee <LTL> \mid <LTL> \ \mathbf{U} <LTL> \\ & \mid <LTL> \ \mathbf{W} <LTL> \end{aligned}$$

In Definition 2, $\mathbf{X}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{U}$, and $\mathbf{W}$ correspond to temporal operators: $\mathbf{X}$ means *next*, $\mathbf{F}$ means *eventually*, $\mathbf{G}$ means *global*, $\mathbf{U}$ means *until*, and $\mathbf{W}$ means *weakly until*. Further, $< LTL >$ may contain propositional variables. In our context, these variables correspond to the execution of activities (e.g. $\mathbf{G}$ (Discharge patient $\Rightarrow$ $\mathbf{F}$ Write discharge letter)).
The temporal operators enable the navigation from point to point on a time line. Definition 3 provides the formal semantics of these temporal operators using recursive equitations.

Definition 3 (Semantics of Linear Temporal Logic). LTL is defined on infinite traces. Hence, for any execution trace $\sigma =< e_1, e_2, e_3, \ldots, e_n >$ we first define its infinite extension $\overline{\sigma} :=< e_1, e_2, e_3, \ldots, e_n, \emptyset, \emptyset, \cdots >$ by adding empty events after event e_n. Further let ϕ and ψ be LTL formulas.

$$\sigma \models \phi :\Leftrightarrow \overline{\sigma} \models \phi$$
$$\overline{\sigma} =< e_1, e_2, e_3, \cdots > \models \mathbf{X}\,\phi :\Leftrightarrow < e_2, e_3, \cdots > \models \phi$$

1. $\overline{\sigma} \models \mathbf{F}\,\phi :\Leftrightarrow \overline{\sigma} \models \phi \vee \mathbf{X}\,\mathbf{F}\,\phi$
2. $\overline{\sigma} \models \mathbf{G}\,\phi :\Leftrightarrow \overline{\sigma} \models \phi \wedge \mathbf{X}\,\mathbf{G}\,\phi$
3. $\overline{\sigma} \models \phi\,\mathbf{U}\,\psi :\Leftrightarrow \overline{\sigma} \models \psi \vee (\phi \wedge \mathbf{X}\,(\phi\,\mathbf{U}\,\psi))$, whereby ψ has to occur eventually (i.e., $\mathbf{F}\,\psi$ holds).
4. $\overline{\sigma} \models \phi\,\mathbf{W}\,\psi :\Leftrightarrow \overline{\sigma} \models \psi \vee (\phi \wedge \mathbf{X}\,(\phi\,\mathbf{W}\,\psi))$, whereby ψ need not occur eventually (i.e., $\mathbf{G}\,\neg\psi$ is allowed).

Example 2 illustrates how LTL can be used for modeling compliance rules.

Example 2 (Modeling Compliance Rules with LTL). Table 2 provides examples illustrating the use of LTL. More precisely, the informal compliance rules from Table 1 are now formally defined based on LTL.

Table 2: Representing the Compliance Rules from Table 1 in LTL

c_1	$\neg$Perform surgery **W** (Prepare patient $\wedge$ ($\neg$Perform surgery **W** Send patient to surgical suite))
c_2	(**G** (Examine patient $\Rightarrow$ **F** Make decision)) $\wedge$ ($\neg$Make decision **U** Examine patient)
c_3	**G** (Examine patient $\Rightarrow$ **F** Inform about risks)
c_4	$\neg$Schedule surgery **W** Inform about anesthesia
c_5	(**G** $\neg$Schedule surgery) $\Rightarrow$ (**G** $\neg$Perform surgery)
c_6	**G** (Discharge patient $\Rightarrow$ **F** Write discharge letter)
c_7	**G** ($\neg$Perform surgery $\Rightarrow$ (**F** Write discharge letter $\Rightarrow$ (($\neg$Write discharge letter **U** Create surgery report) $\wedge$ ($\neg$Write discharge letter **U** Make lab test))))

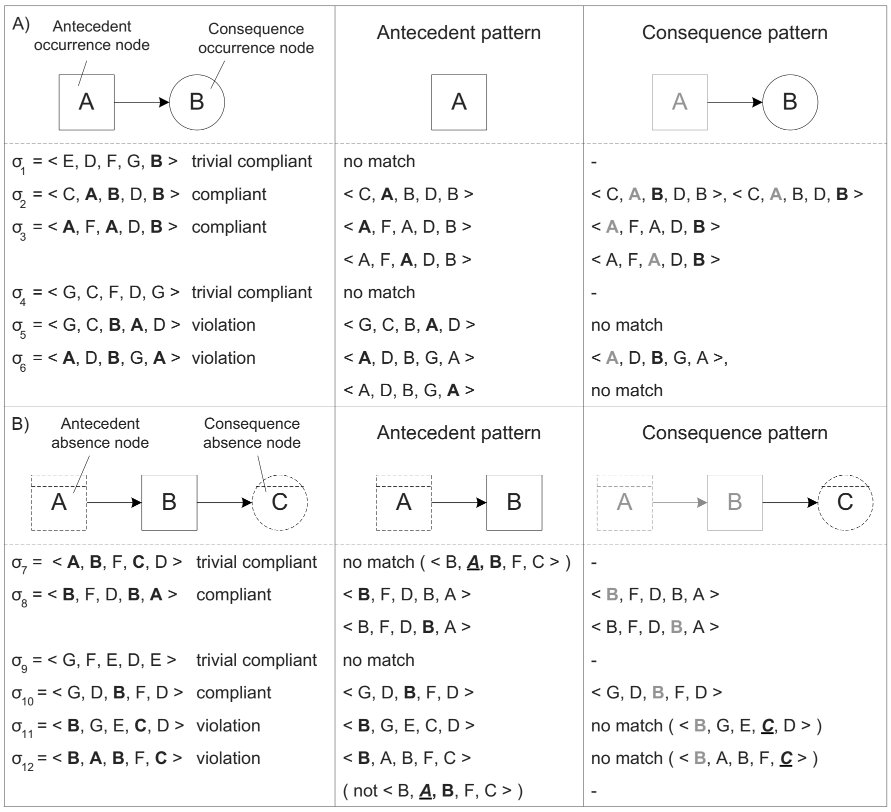
Obviously, the formal definition of compliance rules by the use of LTL or other temporal logics (e.g., Table 2) requires expert knowledge. In particular, LTL expressions will be not understandable to domain experts. Hence, graphical notations like *Compliance Rule Graphs (CRGs)* have been suggested [6]. CRGs allow modeling compliance rules on a higher level of abstraction based on graphs. CRGs further define a compliance rule by means of an *antecedent pattern* complemented by a *consequence pattern*. Both, the antecedent and the consequence pattern consist of occurrence and absence nodes. These nodes are connected by directed edges that may also connect antecedent nodes with consequence nodes. While nodes require the existence or absence of activities, the edges connecting them describe respective activity sequences. Note that edges must not connect two absence nodes.

The semantics of an CRG is as follows: Each trace will be compliant with the CRG, if for any match of the antecedent pattern to the trace's entries the related consequence pattern has to find at least one suitable match as well. Further, if there exists no match of the antecedent pattern the trace will be compliant as well. The latter kind of compliance is denoted as *trivial compliance*.

Any match of the antecedent pattern to a trace is a mapping from each antecedent occurrence node to one of the entries of the trace. For sequenced antecedent occurrence nodes, whose sequence is expressed by edges, the corresponding trace entries have to obey the same sequence. Further, for each antecedent absence node, there must be no trace entry of the antecedent absence node's activity that obeys the sequences with trace entries of adjacent antecedent occurrence nodes denoted by appropriate edges. A suitable match of the consequence pattern maps any consequence occurrence node to a corresponding trace entry as well. Further those trace entries have to consider the sequence denoted by the edges as well. In addition, there must be no trace entry of the consequence absence node's activity that obeys sequences with trace entries of adjacent antecedent and consequence occurrence nodes that are denoted by appropriate edges. Examples 3 and 4 illustrate the semantics of CRG-based constraints.

Example 3 (Compliance of Simple CRGs). We consider Figure 2 in order to exemplarily describe the semantics of CRGs. More precisely, two CRGs and related execution traces are provided in Figure 2A and Figure 2B respectively. Furthermore, for each trace we indicate whether the corresponding process instance complies with the respective CRG or violates it.

Regarding the two CRGs from Figure 2, for example, trivial compliance holds for σ_1, σ_4, and σ_9. Obviously, for each of theses traces at least one antecedent occurrence node can not be mapped to any trace entry; e.g., A does not occur in σ_1. Trace σ_7 constitutes another example of trivial compliance although the antecedent occurrence node B can be mapped to a trace entry; however, trace σ_7 also contains an entry of activity A (preceding the entry of B) which corresponds to the antecedent absence node (i.e., this entry prevents the antecedent pattern from matching with σ_7). To allow for a match of the antecedent pattern in the given context there should not occur an A preceding the B in σ_7.



A)		Antecedent pattern	Consequence pattern
σ_1 = < E, D, F, G, **B** >	trivial compliant	no match	-
σ_2 = < C, **A**, **B**, D, **B** >	compliant	< C, **A**, B, D, B >	< C, A, **B**, D, B >, < C, A, B, D, **B** >
σ_3 = < **A**, F, **A**, D, **B** >	compliant	< **A**, F, A, D, B >	< A, F, A, D, **B** >
		< A, F, **A**, D, B >	< A, F, A, D, **B** >
σ_4 = < G, C, F, D, G >	trivial compliant	no match	-
σ_5 = < G, C, **B**, **A**, D >	violation	< G, C, B, **A**, D >	no match
σ_6 = < **A**, D, **B**, G, **A** >	violation	< **A**, D, B, G, A >	< A, D, **B**, G, A >,
		< A, D, B, G, **A** >	no match
B)		**Antecedent pattern**	**Consequence pattern**
σ_7 = < **A**, **B**, F, **C**, D >	trivial compliant	no match (< B, **A**, **B**, F, C >)	-
σ_8 = < **B**, F, D, **B**, **A** >	compliant	< **B**, F, D, B, A >	< B, F, D, B, A >
		< B, F, D, **B**, A >	< B, F, D, B, A >
σ_9 = < G, F, E, D, E >	trivial compliant	no match	-
σ_{10} = < G, D, **B**, F, D >	compliant	< G, D, **B**, F, D >	< G, D, B, F, D >
σ_{11} = < **B**, G, E, **C**, D >	violation	< **B**, G, E, C, D >	no match (< B, G, E, **C**, D >)
σ_{12} = < **B**, **A**, **B**, F, **C** >	violation	< **B**, A, B, F, C >	no match (< B, A, B, F, **C** >)
		(not < B, **A**, **B**, F, C >)	-

Fig. 2: Simple Compliance Rule Graphs

Consider now the non-trivial compliant traces: $\sigma_2, \sigma_3, \sigma_8$, and σ_{10}. Concerning σ_2, the antecedent pattern A matches once, and there are two suitable matches of the consequence pattern B. Regarding σ_3, A occurs twice. Since B succeeds both occurrences of A, there is a suitable mapping of the consequence pattern in both cases. The same applies to σ_8 and the CRG depicted in Figure 2B: There are two mappings of the antecedent pattern in terms of the two B that do not have a preceding A (but a succeeding one). Further, for both mappings there is no C succeeding the B. Hence, trace σ_8 is compliant with the CRG depicted in Figure 2B. Finally, σ_{10} contains exactly one mapping of the antecedent pattern B. Since no C is following, the consequence pattern maps as well.

Finally, let us consider the non-compliant traces $\sigma_5, \sigma_6, \sigma_{11}$, and σ_{12}. σ_5 violates the CRG from Figure 2A since the antecedent pattern maps on the A, but no suitable mapping of the consequence pattern with a B following the A can be found (the only occurring B precedes A). Regarding σ_6, the antecedent pattern maps twice. However, while for the first A there exists a suitable mapping of the consequence pattern with the B, the second A is not followed by any B; i.e., trace σ_6 violates the CRG depicted in Figure 2A. Regarding the CRG from Figure 2B and σ_{11}, the B allows for the antecedent pattern to match, while the succeeding C prohibits the consequence pattern to match. Finally, consider the violation of the CRG from Figure 2B by σ_{12}: Due to the presence of the A, the antecedence pattern cannot map to the second occurrence of B, but only to the first one. Due to the presence of the C at the end of the trace, however, no suitable match of the consequence pattern is possible.

Example 4 (Compliance of Complex CRGs). Figure 3 provides two additional CRGs and related execution traces. Again, for each trace it is indicated whether the corresponding process instance complies with the respective CRG or violates it.

Regarding Figure 3A, for example, trivial compliance holds for σ_{13} and σ_{16}. Obviously, for each of theses traces at least one antecedent occurrence node can not be mapped to any trace entry. Furthermore, σ_{15}, σ_{21},

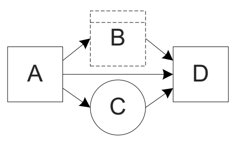
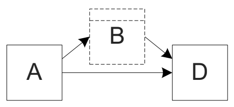
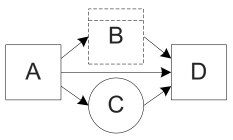
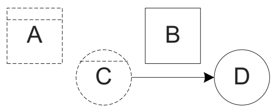
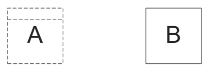
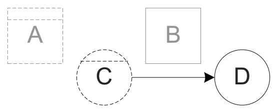
C)	Antecedent pattern	Consequence pattern
		
σ_{13} = < **A**, C, **B**, G, C > trivial compliant	no match	-
σ_{14} = < E, **A**, E, **C**, **D** > compliant	< E, **A**, E, C, **D** >	< E, A, E, **C**, D >
σ_{15} = < E, **A**, E, **B**, **D** > trivial compliant	no match (< E, **A**, E, **B**, **D** >)	-
σ_{16} = < G, **B**, E, G, **D**> trivial compliant	no match	-
σ_{17} = < **A**, F, **D**, G, **B** > violation	< **A**, F, **D**, G, B >	no match
σ_{18} = < **A**, F, **D**, **C**, **D** > violation	< **A**, F, D, C, **D** >	< A, F, D, **C**, D >
	< **A**, F, **D**, C, D >	no match
D)	Antecedent pattern	Consequence pattern
		
σ_{19} = < E, **D**, F, G, **B** > compliant	< E, D, F, G, **B** >	< E, **D**, F, G, B >
σ_{20} = < **D**, F, **C**, E, **B** > compliant	< D, F, C, E, **B** >	< **D**, F, C, E, B >
σ_{21} = < **A, B**, **C**, E, **D** > trivial compliant	no match (< **A**, **B**, C, E, D >)	-
σ_{22} = < G, **C**, **B**, **A**, **D** > trivial compliant	no match (< G, C, **B**, **A**, D >)	-
σ_{23} = < **C**, F, **B**, G, E > violation	< C, F, **B**, G, E >	no match
σ_{24} = < **C**, F, **D**, E, **B** > violation	< C, F, D, E, **B** >	no match (< **C**, F, **D**, E, B >)

Fig. 3: More Complex Compliance Rule Graphs

and σ_{22} also constitute examples of trivial compliance although the antecedent occurrence nodes can be mapped to trace entries; however, the traces contain entries of the antecedent absence nodes' activities as well (i.e., those prevent the antecedent patterns from being matched). Regarding σ_{15} there should be no B between A and D to allow for a match of the antecedent pattern of the CRG from Figure 3A. Regarding σ_{21} and σ_{22} no A should occur, in turn, to allow for a match of the antecedent pattern of the CRG from Figure 3B.

Consider now the non-trivial compliant traces σ_{14}, σ_{19}, and σ_{20}. σ_{14} contains an A succeeded by a D; between these entries there is no B such that the antecedent pattern of respective CRG (cf. Figure 3A) matches. Furthermore, the consequence pattern also matches since σ_{14} contains an entry of the required C between A and D. With a B and no A the two traces σ_{19} and σ_{20} allow for mappings of the antecedent pattern. Further, both traces contain a D not preceded by C (while C in σ_{20} succeeds the D, σ_{19} contains no C at all); i.e., both traces allow for a suitable mapping of the consequence pattern, and are thus compliant with the CRG from Figure 3B.

Finally, let us consider the non-compliant traces $\sigma_{17}, \sigma_{18}, \sigma_{23}$, and σ_{24}. Regarding Figure 3A and trace σ_{17}, the antecedence pattern can be mapped to the trace entries A and D, since the B is not in between; however, the consequence pattern cannot match since σ_{17} contains no C. Trace σ_{18} even enables two matches of the antecedent pattern of the CRG from Figure 3A: the first one consists of the A and the D in the middle, while the second match consists of the same A and the D at the end. Since the latter is preceded by C, the second match can be enriched with a suitable mapping of the consequence pattern. Nevertheless, trace σ_{18} violates the CRG from Figure 3A, since there is no C between the A and the D of the first mapping. Regarding trace σ_{23}, the antecedent pattern maps to the B, but the D of the consequence pattern is missing (i.e., the C does not matter). Indeed, σ_{24} even contains a D, but this is preceded by a C; i.e., the consequence pattern cannot map while the antecedent pattern maps. Hence, σ_{24} violates the CRG depicted in Figure 3B.

Example 5 (Modeling Compliance Rules by the Use of CRGs). In Figure 4, the compliance rules from Table 1 and Table 2 respectively are re-modeled by means of CRGs.

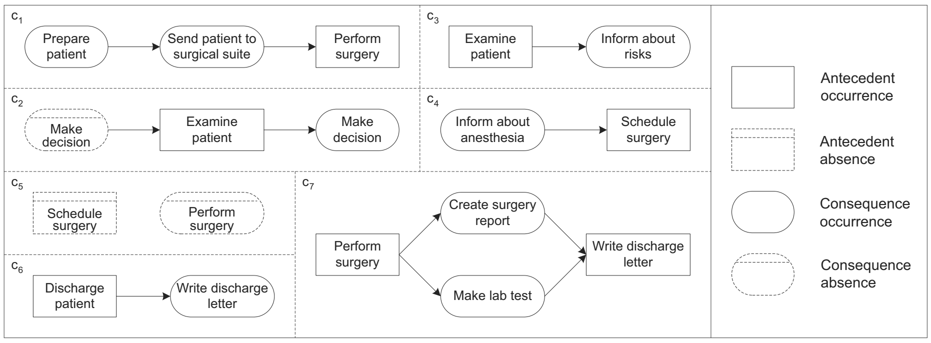


Fig. 4: Representing the Compliance Rules from Tables 1 and 2 as CRGs

3 A-priori Compliance Checking

Once the compliance rules have been modeled (e.g., by using CRGs), compliance of pre-specified process models with those rules can be checked. This is denoted as *a-priori compliance checking* since the compliance of processes with regulations is checked before their execution, i.e., before any process instance is executed based on the pre-specified process model. According to Definition 4, a pre-specified process model *totally complies* with a given compliance rule, if and only if the model solely allows for traces being compliant with the rule. Further, we define the notions of *partial compliance* and *partial violation* as well as *total violation*.

Definition 4 (Compliance of Pre-specified Process Model). Let S be a pre-specified process model and let ϕ be a compliance rule (cf. Definition 1). Further, let $QS_S \subseteq \Sigma^\star$ be the set of all complete traces producible on S; i.e., $\sigma \in QS_S$ represents a completed process instance. Then:

- *S (totally) complies* with ϕ, if and only if all complete traces σ being producible on S comply with ϕ; i.e., $\forall \sigma \in QS_S : \phi(\sigma)$.
- *S partially complies* with ϕ, if and only if there exists a complete trace σ being producible on S and complying with ϕ; i.e., $\exists \sigma \in QS_S : \phi(\sigma)$
- *S partially violates* ϕ, if and only if there exists a complete trace σ being producible on S and violating ϕ; i.e., $\exists \sigma \in QS_S : \neg\phi(\sigma)$.
- *S only partially complies* with ϕ, if and only if S partially complies with ϕ as well as S partially violates ϕ; i.e., $\exists \sigma_1, \sigma_2 \in QS_S : \phi(\sigma_1) \wedge \neg\phi(\sigma_2)$
- *S (totally) violates* ϕ, if and only if all complete traces σ being producible on S violate the compliance rule ϕ; i.e., $\forall \sigma \in QS_S : \neg\phi(\sigma)$.

In case S totally complies with ϕ, for brevity we also use the phrase "*S complies* with ϕ". The same applies if S totally violates ϕ. In this case we also say "*S violates ϕ*".

Example 6 illustrates the different notions.

Example 6 (Compliance of a Pre-specified Process Model). Reconsider the pre-specified process model S_{med} from Figure 1 and the compliance rules from Table 1 and Fig. 4 respectively. Process model S_{med} *(totally) complies* with compliance rules c_2, c_5, c_6, and c_7. It *only partially complies* with compliance rules c_3 and c_4, while compliance rule c_1 is *(totally) violated.*

One common way to perform *a priori* checking is the usage of model checking techniques [5]. These allow for verifying models and systems against temporal formulas. In this context tools exist that provide efficient implementations of model checking algorithms. Generally, one can distinguish between *explicit model checking* and *symbolic model checking* . In the context of LTL, explicit model checking means to first create a state-based automaton that represents the negated formula. Then, this automaton and the state space of the process model are explored in combination. Symbolic model checking, in term, transforms both the process model and the compliance rule into propositional logic expressions and then applies a satisfiability check. When applying model checking to the verification of compliance rules not being modeled in terms of temporal logic (e.g., compliance rules that are modeled based on CRGs), these rules first have to be transformed into temporal logic.

4 Compliance Monitoring

Checking business process compliance of a pre-specified process model *a priori* at build-time is not always feasible, e.g., if the process model is too large or compliance rules are too complex or depend on run-time data. Besides, loosely specified and dynamically evolving processes require support for ensuring compliance during run-time as well. Hence, *compliance monitoring* is required that allows process engineers to control and monitor compliance rules during the execution of single process instances. However, at the process instance level it is not sufficient to only consider one snapshot, i.e. to state whether or not the process instance violates a particular compliance rule at a certain point in time. On the one hand, the violation of a certain compliance rule can often be cured later on when the process instance progresses. On the other hand, there are violations for which no adequate continuation exists. Hence, Definition 5 not only distinguishes between process instances being compliant and those violating a compliance rule, but also between curable and incurable violations of process instances regarding an imposed compliance rule.

Definition 5 (Compliance and Curability of Process Instances). Let I be a process instance represented by its current trace σ_I. Further, the process model based on which I has been executed may not be known. Finally, let ϕ be a compliance rule. Then:

- *I complies* with ϕ, if and only if σ_I complies with ϕ; i.e., $\phi(\sigma_I)$.
- *I violates* ϕ, if and only if holds σ_I violates ϕ; i.e., $\neg\phi(\sigma_I)$.
- *I curably violates* ϕ, if and only ifσ_I violates ϕ, but the execution of I can be continued in such a way that the resulting trace complies with ϕ; i.e., $\neg\phi(\sigma_I) \wedge \exists\tau \in \Sigma^\star : \phi(\sigma_I\tau)$.
- *I incurably violates* ϕ, if and only if σ_I violates ϕ and any continuation of I violates ϕ as well; i.e., $\neg\phi(\sigma_I) \wedge \forall\tau \in \Sigma^\star : \neg\phi(\sigma_I\tau)$.

Example 7 illustrates Definition 5.

Example 7 (Compliance and Curability of Process Instances). Consider the compliance rules c_2, c_3 and c_4 from Table 1 (see also Table 2 and Figure 4). Further, consider the traces σ_{I_1} and σ_{I_2} of the running process instances I_1 and I_2 respectively (cf. Figure 5). Obviously, I_1 violates c_2, while it complies with c_3 and c_4. Further, c_2 is curably violated, since σ_{I_1} can be continued by executing activity *Make decision.*

Finally, I_2 complies with c_2 and c_3. However, I_2 incurably violates c_4 since no continuation of σ_{I_2} contains the activity *Inform about anesthesia* preceding *Schedule surgery.*

σ_{I_1}	σ_{I_2}
1 Admit patient	1 Admit patient
2 Perform checkup	2 Perform checkup
3 Examine patient	3 Examine patient
4 Inform about risks	4 Inform about risks
	5 Make decision
	6 Schedule surgery

Fig. 5: Snapshots of Instance Traces

In practice, it is not always feasible to only deploy process models being totally compliant; i.e., there may be pre-specified process models that only partially comply with imposed compliance rules. As will be shown in Example 8, instances of respective pre-specified process model need to be monitored at run-time to determine whether or not a compliance violation can be cured in the following. According to this, Definition 6 distinguishes between different levels of criticality of curable violations.

Definition 6 (Temporary and Permanent Compliance Violations). Let $I = (S, \sigma_I)$ be a process instance running on a process model S. Further, let QS_S be the set of all complete traces producible on S and ϕ be a compliance rule. Then:

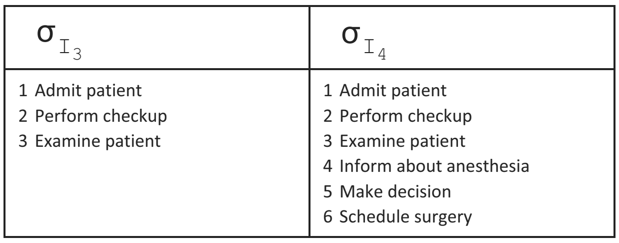
- *I temporarily violates ϕ*, if and only if I currently violates ϕ, but any continuation on S will comply with ϕ at least at one future point in time:
$$I \text{ curably violates } \phi \wedge \forall \tau \in \Sigma^\star \text{ with } \sigma_I\tau \in QS_S : \\ \exists \upsilon, \omega \in \Sigma^\star \text{ with } \upsilon\omega = \tau \wedge \phi(\sigma_I \upsilon).$$
- *I potentially violates ϕ temporarily*, if and only if I currently violates ϕ and it holds: On the one hand, I may be continued in a way such that it will comply with ϕ at least at one future point in time. On the other hand, I may be also continued in a way such that it will never comply with ϕ again; i.e., I curably violates $\phi \wedge \exists \tau_1, \tau_2 \in \Sigma^\star$: for $\sigma_I\tau_1, \sigma_I\tau_2 \in QS_S$ it holds:
$$(\exists \upsilon_1, \omega_1 \in \Sigma^\star \text{ with } \upsilon_1\omega_1 = \tau_1 : \phi(\sigma_I\upsilon_1)) \;\wedge\; (\forall \upsilon_2, \omega_2 \in \Sigma^\star \text{ with } \upsilon_2\omega_2 = \tau_2 : \neg\phi(\sigma_I\upsilon_2)).$$
- *I permanently violates ϕ*, if and only if I currently violates ϕ and any continuation on S always violates ϕ; i.e.,
$$I \text{ curably violates } \phi \wedge \forall \tau \in \Sigma^\star \text{ with } \sigma_I\tau \in QS_S: \\ \forall \upsilon, \omega \in \Sigma^\star \text{ with } \upsilon\omega = \tau : \neg\phi(\sigma_I\upsilon).$$

Example 8 applies Definition 6 to selected process instances.

Example 8 (Persistence of Compliance Violations). Reconsider the compliance rules c_2, c_3 and c_4 from Table 1 (see also Table 2 and Figure 4). Further consider the traces σ_{I_3} and σ_{I_4} from Figure 6. These correspond to the running process instances $I_3 = (S_{med}, \sigma_{I_3})$ and $I_4 = (S_{med}, \sigma_{I_4})$, which are executed on the pre-specified process model S_{med} from Figure 1.

- Obviously, I_3 violates c_2 and c_3, while it complies with c_4. Further, c_2 is only temporarily violated by I_3, since its continuation on S_{med} will lead to the execution of *Make decision* (e.g., σ_{I_2} and σ_{I_4}). However, c_3 is potentially temporarily violated, since S_{med} allows σ_{I_3} continuing with activity *Inform about risks* (e.g., σ_{I_1} and σ_{I_2}) or without activity *Inform about risks* (e.g. σ_{I_4}).

- I_4 violates c_3, but complies with c_2 and c_4. Further, c_3 is permanently violated by I_4, since no continuation of I_4 on S_{med} will contain the required activity *Inform about risks*.

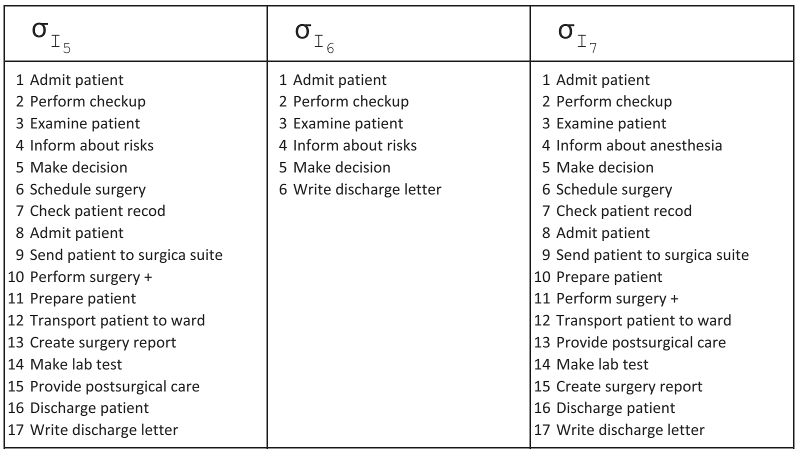


σ_{I_3}	σ_{I_4}
1 Admit patient	1 Admit patient
2 Perform checkup	2 Perform checkup
3 Examine patient	3 Examine patient
	4 Inform about anesthesia
	5 Make decision
	6 Schedule surgery

Fig. 6: Additional Snapshots of Process Instance Traces

5 A-posteriori Compliance Checking

Instead of ensuring compliance *a priori* (i.e., by checking pre-specified process models at build-time) or monitoring it during processes execution, compliance may be also checked for completed process instances *a-posteriori*; e.g., to determine whether these completed instances comply with newly emerging regulations. Compliance of completed process instances can be directly decided based on the definition of compliance rules (cf. Definition 1).



σ_{I_5}	σ_{I_6}	σ_{I_7}
1 Admit patient	1 Admit patient	1 Admit patient
2 Perform checkup	2 Perform checkup	2 Perform checkup
3 Examine patient	3 Examine patient	3 Examine patient
4 Inform about risks	4 Inform about risks	4 Inform about anesthesia
5 Make decision	5 Make decision	5 Make decision
6 Schedule surgery	6 Write discharge letter	6 Schedule surgery
7 Check patient recod		7 Check patient recod
8 Admit patient		8 Admit patient
9 Send patient to surgica suite		9 Send patient to surgica suite
10 Perform surgery +		10 Prepare patient
11 Prepare patient		11 Perform surgery +
12 Transport patient to ward		12 Transport patient to ward
13 Create surgery report		13 Provide postsurgical care
14 Make lab test		14 Make lab test
15 Provide postsurgical care		15 Create surgery report
16 Discharge patient		16 Discharge patient
17 Write discharge letter		17 Write discharge letter

Fig. 7: Execution Traces of Completed Process Instances

Example 9 illustrates *a-posteriori* compliance checking.

Example 9 (Compliance of Process Execution Logs). Consider the compliance rules c_1, c_2, c_3, c_4, c_5, c_6, and c_7 from Table 1 (see also Table 2 and Figure 4). Further consider the execution traces σ_{I_5}, σ_{I_6} and σ_{I_7} from Figure 7, which correspond to the completed process instances I_5, I_6 and I_7. I_5 violates c_1 and c_4, and complies with c_2, c_5, c_6, and c_7. Further, I_6 complies with c_i, $i = 1 \ldots 7$ and I_7 violates c_1 and c_3, but complies with c_2, c_4, c_5, c_6, and c_7.

Similar to *a-priori* compliance checking, *a-posteriori* compliance checking can be realized based on techniques that build on model checking. The approach described in [7] transforms LTL-based compliance rules into state-based automata. Taking an execution log as input, these automata allow deciding whether a completed process instance complies with the original rule or violates it.

6 Effects of Process Changes on Compliance

As discussed in [8, 9], pre-specified process models as well as process instances running on them may have to be changed and adapted. Obviously, such changes can affect compliance of the process models and process instances, respectively, with the imposed compliance rules. Depending on theses effects, we define compliance of changes with a given compliance rule (cf. Definition 7).

Definition 7 (Compliance of Changes). Let S be a pre-specified process model and let $I = (S, \sigma_I)$ be a related process instance. Further, let Δ be a change that correctly transforms the pre-specified process model S into another pre-specified process model S'. Finally, let $I = (S, \sigma_I)$ be correctly migratable to S', i.e., $I = (S', \sigma_I)$. Then:

- The application of Δ to S meets compliance rule ϕ, if and only if the application of Δ to S does not weaken the compliance of S with ϕ; i.e.,
  - S complies with $\phi \Rightarrow S'$ complies with ϕ.
  - S partially complies with $\phi \Rightarrow S'$ partially complies with ϕ.
- The application of Δ to $I = (S, \sigma_I)$ meets compliance rule ϕ, if and only if the application of Δ to process instance I does not weaken the compliance of I with ϕ; i.e.,
  - $I = (S, \sigma_I)$ complies with $\phi \Rightarrow (S', \sigma_I)$ complies with ϕ.
  - $I = (S, \sigma_I)$ temporarily violates $\phi \Rightarrow ((S', \sigma_I)$ temporarily violates $\phi \vee (S', \sigma_I)$ complies with $\phi)$.
  - $I = (S, \sigma_I)$ potentially violates ϕ temporarily $\Rightarrow ((S', \sigma_I)$ potentially violates ϕ temporarily
    $\vee (S', \sigma_I)$ temporarily violates $\phi \vee (S', \sigma_I)$ complies with $\phi)$.

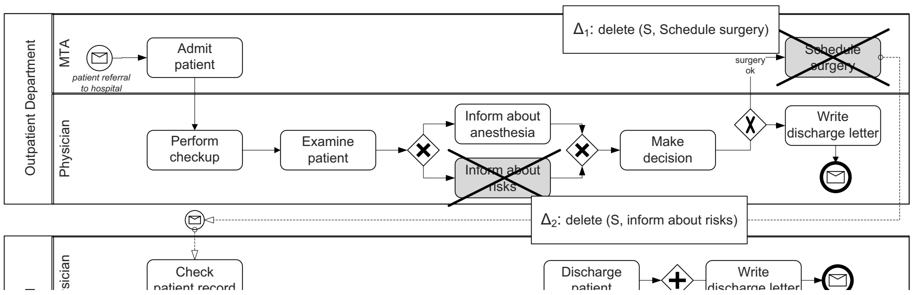
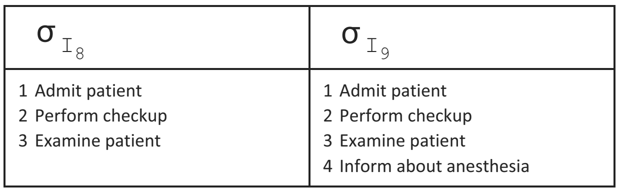


Fig. 8: Changes Potentially Affecting the Compliance of Process Model S_{med}



σ_{I_8}	σ_{I_9}
1 Admit patient 2 Perform checkup 3 Examine patient	1 Admit patient 2 Perform checkup 3 Examine patient 4 Inform about anesthesia

Fig. 9: Further Examples for Snapshots of Process Instance Traces

When applying Definition 7 in a straightforward manner, one would have to re-check compliance of a process model with all defined compliance rules whenever changing this model. This might become necessary in the context of ad-hoc adaptations of single process instances or changes of a pre-specified process models solely at the process type level (i.e., without propagating the type change to already running process instances). However, re-checking business compliance for large collections of running process instances might be too expensive. More precisely, for each of these hundreds up to thousands of process instances it has to be determined whether or not it still meets the imposed compliance rules when migrating the process instance to the new process model version. To cope with this challenge, changes and compliance rules have to be analyzed (e.g., by considering the affected activities) in order to restrict the set of compliance rules to be re-checked.

Example 10 (Effects of Changes on Process Model Compliance). Take compliance rules c_4 and c_5 from Table 1 (see also Table 2 and Figure 4) and consider change Δ_1 of the pre-specified process model S_{med} as depicted in Figure 8. Obviously, Δ_1 meets c_4. While S only partially complies with c_4, S' totally complies with this rule. By contrast, Δ_1 violates c_5 since S totally complies with c_5, but S' only partially complies with this rule.

Example 11 (Effects of Changes on Process Instance Compliance). Consider now compliance rule c_3 from Table 1 (see also Table 2 and Figure 4) and change Δ_2 from Figure 8 that transforms S_{med} into S'_{med}. Further, consider the process instances $I_8 = (S_{med}, \sigma_{I_8})$ and $I_9 = (S_{med}, \sigma_{I_9})$ from Figure 9 that both depend on the pre-specified process model S_{med} from Figure 1. Regarding I_8, Δ_2 violates c_3: $I_8 = (S_{med}, \sigma_{I_8})$ potentially violates c_3 temporarily, whereas (S'_{med}, σ_{I_8}) permanently violates this rule. However, regarding I_9, Δ_2 meets c_3 since $I_9 = (S_{med}, \sigma_{I_9})$ permanently violates c_3 which also applies to (S'_{med}, σ_{I_9}) permanently.

7 User Perspective

This section gives an idea how compliance checking looks like from the perspective of the user. Currently, only few tools exist that allow ensuring business process compliance at the process type or the process instance level. One of them is the SeaFlows Toolset [10], which provides a comprehensive and extensible framework for checking business compliance of pre-specified process models. For this purpose, the SeaFlows Toolset provides a user-friendly environment. For modeling compliance rules SeaFlows uses CRGs as presented in Section 2.

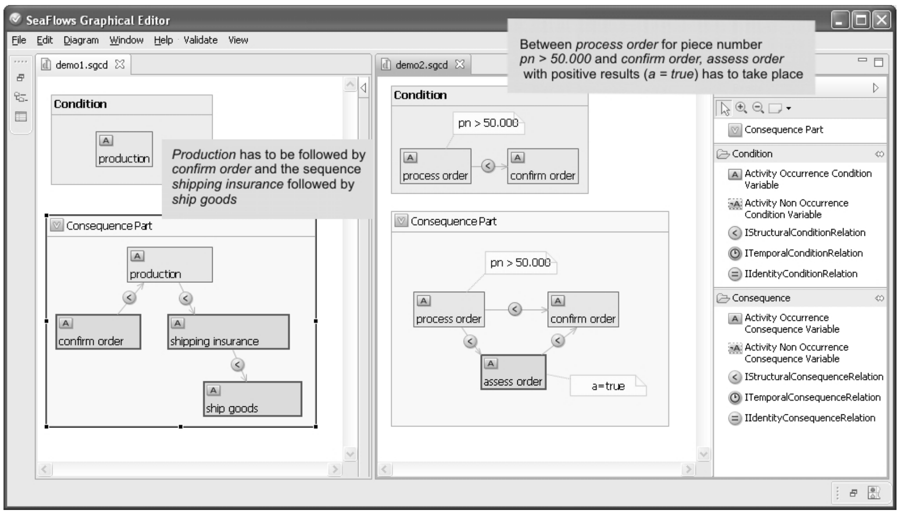
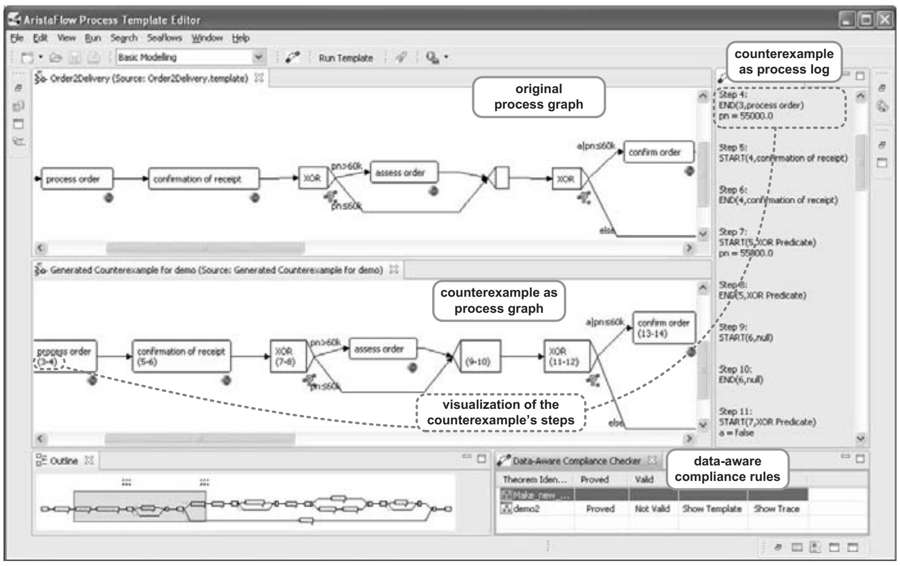

SeaFlows Graphical Editor
File Edit Diagram Window Help Validate View
demo1.sgcd
Condition
production
Production has to be followed by confirm order and the sequence shipping insurance followed by ship goods
Consequence Part
production
confirm order
shipping insurance
ship goods
demo2.sgcd
Between process order for piece number pn > 50.000 and confirm order, assess order with positive results (a = true) has to take place
Condition
pn > 50.000
process order
confirm order
Consequence Part
pn > 50.000
process order
confirm order
assess order
a=true
Consequence Part
Condition
Activity Occurrence Condition Variable
Activity Non Occurrence Condition Variable
IStructuralConditionRelation
ITemporalConditionRelation
IIdentityConditionRelation
Consequence
Activity Occurrence Consequence Variable
Activity Non Occurrence Consequence Variable
IStructuralConsequenceRelation
ITemporalConsequenceRelation
IIdentityConsequenceRelation

Fig. 10: Modeling Compliance Rules with the SeaFlows Graphical Editor

The SeaFlows Toolset allows enriching process models with these rules and checking for compliance with them. Furthermore, compliance checking considers data as well as efficiency issues by applying a number of abstraction strategies. Finally, violations of compliance rules are illustrated by means of a counter example (cf. Figure 11). At the technical level the applied compliance checking approach uses the model checker SAL [11].
Additionally, a structural compliance checking approach is delivered. It first derives structural criteria from compliance rules. Then it applies those criteria to check business process compliance of cycle-free process models (cf. Figure 12).


AristaFlow Process Template Editor
File Edit View Run Search Seaflows Window Help
Basic Modeling
Run Template
Order2Delivery (Source: Order2Delivery.template)
original process graph
process order
confirmation of receipt
XOR
pn>60k
pn≤60k
assess order
a|pn≤60k
confirm order
else
Generated Counterexample for demo (Source: Generated Counterexample for demo)
counterexample as process graph
process order (3-4)
confirmation of receipt (5-6)
XOR (7-8)
(9-10)
XOR (11-12)
confirm order (13-14)
visualization of the counterexample's steps
counterexample as process log
Outline
Data-Aware Compliance Checker
data-aware compliance rules

Step 4:
END(3,process order)
pn = 55000.0

Step 5:
START(4,confirmation of receipt)

Step 6:
END(4,confirmation of receipt)

Step 7:
START(5,XOR Predicate)
pn = 55000.0

Step 8:
END(5,XOR Predicate)

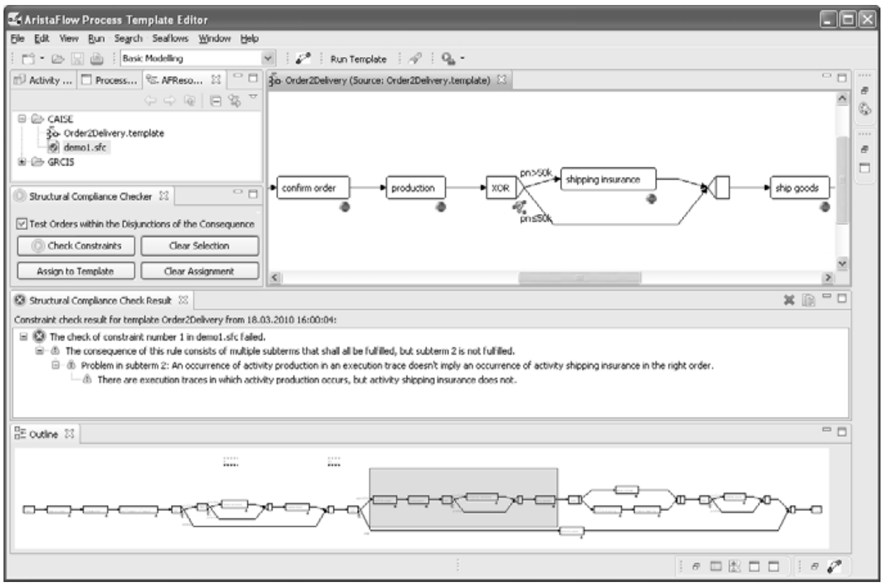
Step 9:
START(6,null)

Step 10:
END(6,null)

Step 11:
START(7,XOR Predicate)
a = false

Theorem Iden...	Proved	Valid		
Make_new_...				
demo2	Proved	Not Valid	Show Template	Show Trace

Fig. 11: Compliance Checking with the SeaFlows Toolset



[✓] Test Orders within the Disjunctions of the Consequence

Constraint check result for template Order2Delivery from 18.03.2010 16:00:04:

- The check of constraint number 1 in demo1.sfc failed.
  - The consequence of this rule consists of multiple subterms that shall all be fulfilled, but subterm 2 is not fulfilled.
    - Problem in subterm 2: An occurrence of activity production in an execution trace doesn't imply an occurrence of activity shipping insurance in the right order.
      - There are execution traces in which activity production occurs, but activity shipping insurance does not.

Fig. 12: Structural Compliance Checking with the SeaFlows Toolset

8 Existing Approaches Enabling Business Process Compliance

Existing approaches enabling business process compliance follow different paradigms to model compliance rules. In first position there are approaches using temporal logic. For example, the work discussed in [12] applies LTL and the one presented in [13] applies CTL for modeling compliance rules. Further, these approaches apply model checking for enabling *a priori* compliance checking. Other logic-based approaches consider the modalities of compliance rules (e.g., obligations or permissions) and use deontic logic as formal basis [14, 15]. As discussed in Section 2, however, logic expressions are less comprehensible to end users. To improve this situation, a pattern-based notation is suggested by Dwyer et al. in [16]. Finally, several approaches use graphical notations (including CRGs) [6, 17, 12].
Model checking is the most common technique for verifying compliance rules (e.g. [13, 17, 12, 18]). However, model checking depends on the exploration of the state space of pre-specified process models. In particular, the state space explosion problem constitutes a big obstacle for the practical use of model checking techniques. To tackle this challenge, techniques like graph reduction and sequentialization of parallel flows as well as predicate abstraction are applied [12, 17, 19]. Besides model checking, there exist other techniques ensuring business process compliance *a priori*. For cycle-free process models, for instance, [20] and [21] provide efficient algorithms.

Generally, compliance rules should not be restricted to the behavior perspective, but be applicable to other perspectives of a PAIS as well (e.g., the information or time perspectives). Compliance checking of process models having state-based data objects (i.e., enumerations), for instance, is suggested by Awad et al. [22]. Further, [19] enables data-aware compliance checking for larger data types (e.g., integers or reals). The verification of cycle-free process models against temporal compliance rules is addressed by Eder et al. [23], while [18] considers both the information and the time perspective.

9 Summary

This report dealt with issues related to business process compliance. Compliance can be checked *a-priori* for pre-specified process models as well as for running process instances or completed ones (i.e., execution logs). For each of these artifacts it can be verified whether or not it complies with compliance rules imposed from regulations, laws and guidelines. This report presented two ways for modeling compliance rules: LTL and CRGs. It first discussed how to apply *a-priori* compliance checking to pre-specified process models and then gave insights into compliance monitoring and different kinds of compliance violations including compliance checking. Following this, it discussed the potential impact of process changes (at both the type and the instance level) on business process compliance. Finally, the report discussed the user perspective as well as recent approaches enabling business process compliance.

References

1. Weske, M.: Workflow management systems: Formal foundation, conceptual design, implementation aspects. Habilitation Thesis, University of Münster, Germany. Springer (2007)
2. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1) (1998) 21–66
3. Weber, B., Reichert, M., Rinderle-Ma, S., Wild, W.: Providing integrated life cycle support in process-aware information systems. Int. Journal of Cooperative Information Systems (IJCIS) **18**(1) (2009) 115–165
4. Reichert, M., Dadam, P.: ADEPT$_{flex}$ – supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems **10**(2) (1998) 93–129
5. Huth, M., Ryan, M.: Logic in Computer Science: Modelling and reasoning about systems. Cambridge University Press (2004)
6. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Proc. 22nd Int. Conf. Advanced Systems Engineering (CAiSE'10), Springer (2010) 9–23
7. van der Aalst, W.M.P., de Beer, H., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: Proc. 13th Int. Conf. Coop. Inf. Systems (CoopIS'05), Springer (2005) 130–147
8. Weber, B., Sadiq, S., Reichert, M.: Beyond rigidity – dynamic process lifecycle support. Computer Science – Research and Development **23**(2) (2009) 47–65
9. Reichert, M., Rinderle-Ma, S., Dadam, P.: Flexibility in process-aware information systems. Transactions on Petri Nets and Other Models of Concurrency II **2** (2009) 115–135
10. Ly, L., 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: Proc. CAISE'10 Forum - Information Systems Evolution. (2011) 76–91
11. Bensalem, S., et al.: An overview of SAL. In: Proc. of the 5th NASA Langley Formal Methods Workshop, NASA Langley Research Center (2000) 187–196
12. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Proc. 6th Int. Conf. Business Process Management (BPM'08), Springer (2008) 326–341
13. Ghose, A.K., Koliadis, G.: Auditing business process compliance. In: Proc. 5th Int. Conf. Service-Oriented Computing (ICSOC'07), Springer (2007) 169–180
14. Alberti, M., et al.: Expressing and verifying business contracts with abductive logic programming. In: Proc. 2nd Int. Conf. Normative Multi-agent Systems (NorMAS'07). Dagstuhl Seminar Proceedings (2007)
15. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: Proc. BPM'06 Workshops, Springer (2006) 5–14
16. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Property specification patterns for finite-state verification. In: Proc. 2nd Workshop Formal Methods in Software Practice (FMSP'98), ACM (1998)
17. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal **46**(2) (2007) 335–261
18. Kokash, N., Krause, C., de Vink, E.: Time and data aware analysis of graphical service models. In: Proc. 8th Int. Conf. Software Engineering and Formal Methods (SEFM'10), IEEE Computer Society (2010)
19. Knuplesch, D., Ly, L., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: Proc. 29th Int. Conf. Conceptual Modeling (ER'2010), Springer (2010)
20. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Proc. 10th Int. Enterprise Distributed Object Computing Conf. (EDOC'06), IEEE Computer Society (2006) 221–232

21. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proc. 3rd Int. workshop on Semantic Business Process Management (SBPM'08). (2008)
22. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for dataaware compliance rules. In: Proc. of 7th Int. Conf. Service Oriented Computing (ICSOC'09), Springer (2009) 500–515
23. Eder, J., Tahamtan, A.: Temporal conformance of federated choreographies. In: Proc. 19th Int. Conf. Database and Expert Sys. App. (DEXA'08), Springer (2008) 668–675