Ensuring Business Process Compliance
Along the Process Life Cycle

David Knuplesch, Manfred Reichert

Ulmer Informatik-Berichte
Nr. 2011-06
Juli 2011
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 $\Sigma$ be the set of all activities and let $\Sigma^*$ be the set of all possible execution traces of processes based on activities from $\Sigma$. Then: A compliance rule $\phi$ defines a function $\phi : \Sigma^* \rightarrow \text{Boolean}$ that considers any trace $\sigma = \langle e_1, \ldots, e_k > \in \Sigma^*$ either to be true (i.e., to be compliant with $\phi$) or false (i.e. to violate $\phi$ or to be not compliant with it). We further denote $\sigma |$ $\phi$ $\iff \phi(\sigma)$ and say trace $\sigma$ satisfies compliance rule $\phi$.

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.
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

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

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):

$$<LTL>::= \top | \bot | \neg <LTL> | ( <LTL> ) | X <LTL> | F <LTL> | G <LTL> | <LTL> \land <LTL> | <LTL> \Rightarrow <LTL> | <LTL> \lor <LTL> | <LTL> U <LTL> | <LTL> W <LTL>$$

In Definition 2, $X$, $F$, $G$, $U$, and $W$ correspond to temporal operators: $X$ means next, $F$ means eventually, $G$ means global, $U$ means until, and $W$ means weakly until. Further, $<LTL>$ may contain propositional variables. In our context, these variables correspond to the execution of activities (e.g. $G (\text{Discharge patient} \Rightarrow F \text{Write discharge letter})$).

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 $\sigma = <e_1, e_2, e_3, \ldots, e_n, \emptyset, 0, \ldots>$ by adding empty events after event $e_n$. Further let $\phi$ and $\psi$ be LTL formulas.

$$\sigma \models \phi \iff \sigma = <e_1, e_2, e_3, \ldots > \models \phi$$

1. $\sigma = F \phi \iff \sigma = <e_2, e_3, \ldots > \models \phi \lor X \sigma \models \phi$
2. $\sigma = G \phi \iff \sigma = <e_2, e_3, \ldots > \models \phi \land X \sigma \models \phi$
3. $\sigma = \phi U \psi \iff \sigma = <e_2, e_3, \ldots > \models \psi \lor (\phi \land X (\phi U \psi))$, whereby $\psi$ has to occur eventually (i.e., $F \psi$ holds).
4. $\sigma = \phi W \psi \iff \sigma = <e_2, e_3, \ldots > \models \psi \lor (\phi \land X (\phi W \psi))$, whereby $\psi$ need not occur eventually (i.e., $G \neg \psi$ is allowed).

Example 2 illustrates how LTL can be used for modeling compliance rules.
Table 2: Representing the Compliance Rules from Table 1 in LTL

<table>
<thead>
<tr>
<th>Rule</th>
<th>LTL Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$\neg\text{Perform surgery \text{W} (Prepare patient } \land (\neg\text{Perform surgery \text{W} Send patient to surgical suite}))$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$(\text{G (Examine patient } \Rightarrow \text{F Make decision)}) \land (\neg\text{Make decision \text{U Examine patient})}$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$\text{G (Examine patient } \Rightarrow \text{F Inform about risks)}$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$\neg\text{Schedule surgery \text{W Inform about anesthesia}$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$(\text{G ~Schedule surgery ) } \Rightarrow (\text{G ~Perform surgery})$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$(\text{G ~Discharge patient } \Rightarrow \text{F Write discharge letter)}$</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$(\neg\text{Perform surgery } \Rightarrow (\text{F Write discharge letter } \Rightarrow ((\neg\text{Write discharge letter \text{U Create surgery report) } \land (\neg\text{Write discharge letter \text{U Make lab test}))})$</td>
</tr>
</tbody>
</table>

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 $\sigma_1$, $\sigma_4$, and $\sigma_9$. Obviously, for each of these traces at least one antecedent occurrence node can not be mapped to any trace entry; e.g., $A$ does not occur in $\sigma_1$. Trace $\sigma_7$ constitutes another example of trivial compliance although the antecedent occurrence node $B$ can be mapped to a trace entry; however, trace $\sigma_7$ 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 $\sigma_7$). To allow for a match of the antecedent pattern in the given context there should not occur an $A$ preceding the $B$ in $\sigma_7$. 

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.
Consider now the non-trivial compliant traces: $\sigma_2$, $\sigma_3$, $\sigma_8$, and $\sigma_{10}$. Concerning $\sigma_2$, the antecedent pattern $A$ matches once, and there are two suitable matches of the consequence pattern $B$. Regarding $\sigma_3$, $A$ 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 $\sigma_8$ 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 $\sigma_8$ is compliant with the CRG depicted in Figure 2B. Finally, $\sigma_{10}$ 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 $\sigma_{12}$. $\sigma_5$ 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 $\sigma_6$, 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 $\sigma_6$ violates the CRG depicted in Figure 2A. Regarding the CRG from Figure 2B and $\sigma_{11}$, 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 $\sigma_{12}$: 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 $\sigma_{13}$ and $\sigma_{16}$. Obviously, for each of these traces at least one antecedent occurrence node can not be mapped to any trace entry. Furthermore, $\sigma_{15}$, $\sigma_{21}$,
and $\sigma_{22}$ 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 $\sigma_{15}$ 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 $\sigma_{21}$ and $\sigma_{22}$ 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 $\sigma_{14}, \sigma_{19},$ and $\sigma_{20}$. $\sigma_{14}$ 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 $\sigma_{14}$ contains an entry of the required $C$ between $A$ and $D$. With a $B$ and no $A$ the two traces $\sigma_{19}$ and $\sigma_{20}$ also allow for mappings of the antecedent pattern. Further, both traces contain a $D$ not preceded by $C$ (while $C$ in $\sigma_{20}$ succeeds the $D$), $\sigma_{19}$ 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 $\sigma_{24}$. Regarding Figure 3A and trace $\sigma_{17}$, 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 $\sigma_{17}$ contains no $C$. Trace $\sigma_{18}$ 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 $\sigma_{18}$ violates the CRG from Figure 3A, since there is no $C$ between the $A$ and the $D$ of the first mapping. Regarding trace $\sigma_{23}$, the antecedent pattern maps to the $B$, but the $D$ of the consequence pattern is missing (i.e., the $C$ does not matter). Indeed, $\sigma_{24}$ even contains a $D$, but this is preceded by a $C$; i.e., the consequence pattern cannot map while the antecedent pattern maps. Hence, $\sigma_{24}$ violates the CRG depicted in Figure 3B.

![Fig. 3: More Complex Compliance Rule Graphs](image)

<table>
<thead>
<tr>
<th>Antecedent pattern</th>
<th>Consequence pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{1} = \langle A, C, B, G, C \rangle$</td>
<td>$\langle A, A, E, C, D \rangle$</td>
</tr>
<tr>
<td>$\sigma_{2} = \langle A, B, E, B, D \rangle$</td>
<td>$\langle A, E, E, D, D \rangle$</td>
</tr>
<tr>
<td>$\sigma_{3} = \langle A, F, D, G, B \rangle$</td>
<td>$\langle A, F, D, C, D \rangle$</td>
</tr>
<tr>
<td>$\sigma_{4} = \langle A, F, D, C, D \rangle$</td>
<td>$\langle A, F, D, C, D \rangle$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Antecedent pattern</th>
<th>Consequence pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{15} = \langle D, F, C, E, B \rangle$</td>
<td>$\langle D, F, C, E, B \rangle$</td>
</tr>
<tr>
<td>$\sigma_{16} = \langle G, C, B, D \rangle$</td>
<td>$\langle D, F, C, E, B \rangle$</td>
</tr>
<tr>
<td>$\sigma_{17} = \langle C, F, B, G, E \rangle$</td>
<td>$\langle D, F, C, E, B \rangle$</td>
</tr>
<tr>
<td>$\sigma_{18} = \langle C, F, B, E, B \rangle$</td>
<td>$\langle D, F, C, E, B \rangle$</td>
</tr>
</tbody>
</table>
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.

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 $\phi$ be a compliance rule (cf. Definition 1). Further, let $QS_S \subseteq \Sigma^*$ 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 $\phi$, if and only if all complete traces $\sigma$ being producible on $S$ comply with $\phi$; i.e., $\forall \sigma \in QS_S : \phi(\sigma)$.

- $S$ partially complies with $\phi$, if and only if there exists a complete trace $\sigma$ being producible on $S$ and complying with $\phi$; i.e., $\exists \sigma \in QS_S : \phi(\sigma)$

- $S$ partially violates $\phi$, if and only if there exists a complete trace $\sigma$ being producible on $S$ and violating $\phi$; i.e., $\exists \sigma \in QS_S : \neg \phi(\sigma)$.

- $S$ only partially complies with $\phi$, if and only if $S$ partially complies with $\phi$ as well as $S$ partially violates $\phi$; i.e., $\exists \sigma_1, \sigma_2 \in QS_S : \phi(\sigma_1) \land \neg \phi(\sigma_2)$

- $S$ (totally) violates $\phi$, if and only if all complete traces $\sigma$ being producible on $S$ violate the compliance rule $\phi$; i.e., $\forall \sigma \in QS_S : \neg \phi(\sigma)$.

In case $S$ totally complies with $\phi$, for brevity we also use the phrase ”$S$ complies with $\phi$”. The same applies if $S$ totally violates $\phi$. In this case we also say ”$S$ violates $\phi$”.

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 $\sigma_I$. Further, the process model based on which $I$ has been executed may not be known. Finally, let $\phi$ be a compliance rule. Then:

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

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 $\sigma_{I_1}$ and $\sigma_{I_2}$ 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 $\sigma_{I_1}$ 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 \( \sigma_{I_2} \) contains the activity *Inform about anesthesia* preceding *Schedule surgery*.

<table>
<thead>
<tr>
<th>( \sigma_{I_1} )</th>
<th>( \sigma_{I_2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Admit patient</td>
<td>1 Admit patient</td>
</tr>
<tr>
<td>2 Perform checkup</td>
<td>2 Perform checkup</td>
</tr>
<tr>
<td>3 Examine patient</td>
<td>3 Examine patient</td>
</tr>
<tr>
<td>4 Inform about risks</td>
<td>4 Inform about risks</td>
</tr>
<tr>
<td>5 Make decision</td>
<td>5 Make decision</td>
</tr>
<tr>
<td>6 Schedule surgery</td>
<td>6 Schedule surgery</td>
</tr>
</tbody>
</table>

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 \( \phi \) be a compliance rule. Then:

- \( I \) temporarily violates \( \phi \), if and only if \( I \) currently violates \( \phi \), but any continuation on \( S \) will comply with \( \phi \) at least at one future point in time:

  \[
  I \text{ curably violates } \phi \land \forall \tau \in \Sigma^* \text{ with } \sigma_I \tau \in QS_S : \exists v, \omega \in \Sigma^* \text{ with } v \omega = \tau \land \phi(\sigma_I v).
  \]

- \( I \) potentially violates \( \phi \) temporarily, if and only if \( I \) currently violates \( \phi \) and it holds: On the one hand, \( I \) may be continued in a way such that it will comply with \( \phi \) 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 \( \phi \) again; i.e., \( I \) curably violates \( \phi \land \exists \tau_1, \tau_2 \in \Sigma^* : \text{ for } \sigma_I \tau_1, \sigma_I \tau_2 \in QS_S \text{ it holds:} \)

  \[
  (\exists v_1, \omega_1 \in \Sigma^* \text{ with } v_1 \omega_1 = \tau_1 : \phi(\sigma_I v_1)) \land (\forall v_2, \omega_2 \in \Sigma^* \text{ with } v_2 \omega_2 = \tau_2 : \neg \phi(\sigma_I v_2)).
  \]

- \( I \) permanently violates \( \phi \), if and only if \( I \) currently violates \( \phi \) and any continuation on \( S \) always violates \( \phi \); i.e.,

  \[
  I \text{ curably violates } \phi \land \forall \tau \in \Sigma^* \text{ with } \sigma_I \tau \in QS_S : \forall v, \omega \in \Sigma^* \text{ with } v \omega = \tau : \neg \phi(\sigma_I v).
  \]

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 \( \sigma_{I_3} \) and \( \sigma_{I_4} \) 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., \( \sigma_{I_3} \) and \( \sigma_{I_4} \)). However, \( c_3 \) is potentially temporarily violated, since \( S_{med} \) allows \( \sigma_{I_3} \) continuing with activity *Inform about risks* (e.g., \( \sigma_{I_3} \) and \( \sigma_{I_4} \)) or without activity *Inform about risks* (e.g. \( \sigma_{I_4} \)).
- $I_4$ 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.

**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).

**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 $\sigma_{I_5}$, $\sigma_{I_6}$ and $\sigma_{I_7}$ 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 these 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 \( \Delta \) 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 \( \Delta \) to \( S \) meets compliance rule \( \phi \), if and only if the application of \( \Delta \) to \( S \) does not weaken the compliance of \( S \) with \( \phi \); i.e.,
  - \( S \) complies with \( \phi \Rightarrow S' \) complies with \( \phi \).
  - \( S \) partially complies with \( \phi \Rightarrow S' \) partially complies with \( \phi \).

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

**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 $\Delta_1$ of the pre-specified process model $S_{med}$ as depicted in Figure 8. Obviously, $\Delta_1$ meets $c_4$. While $S$ only partially complies with $c_4$, $S'$ totally complies with this rule. By contrast, $\Delta_1$ 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 $\Delta_2$ 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 $I_8$, $\Delta_2$ violates $c_3$: $I_8 = (S_{med}, \sigma_{I_8})$ potentially violates $c_3$ temporarily, whereas $(S'_{med}, \sigma_{I_8})$ permanently violates this rule. However, regarding $I_9$, $\Delta_2$ meets $c_3$ since $I_9 = (S_{med}, \sigma_{I_9})$ permanently violates $c_3$ which also applies to $(S'_{med}, \sigma_{I_9})$ 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.

![Fig. 10: Modeling Compliance Rules with the SeaFlows Graphical Editor](image)

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).
Ensuring Business Process Compliance Along the Process Life Cycle

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 \textit{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 \textit{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


List of technical reports published by the University of Ulm
Some of them are available by FTP from ftp.informatik.uni-ulm.de
Reports marked with * are out of print

91-01  Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe
        Instance Complexity

91-02* K. Gladitz, H. Fassbender, H. Vogler
        Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03* Alfons Geser
        Relative Termination

91-04* J. Köbler, U. Schöning, J. Toran
        Graph Isomorphism is low for PP

91-05 Johannes Köbler, Thomas Thierauf
        Complexity Restricted Advice Functions

91-06* Uwe Schöning
        Recent Highlights in Structural Complexity Theory

91-07* F. Green, J. Köbler, J. Toran
        The Power of Middle Bit

91-08* V. Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara,
        U. Schöning, R. Silvestri, T. Thierauf
        Reductions for Sets of Low Information Content

92-01* Vikraman Arvind, Johannes Köbler, Martin Mundhenk
        On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02* Thomas Noll, Heiko Vogler
        Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03 Fakultät für Informatik
        17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04* V. Arvind, J. Köbler, M. Mundhenk
        Lowness and the Complexity of Sparse and Tally Descriptions

92-05* Johannes Köbler
        Locating P/poly Optimally in the Extended Low Hierarchy

92-06* Armin Künnemann, Heiko Vogler
        Synthesized and inherited functions -a new computational model for syntax-directed semantics

92-07* Heinz Fassbender, Heiko Vogler
        A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing
92-08* Uwe Schöning
On Random Reductions from Sparse Sets to Tally Sets

92-09* Hermann von Hasseln, Laura Martignon
Consistency in Stochastic Network

92-10 Michael Schmitt
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 Johannes Köbler, Seinosuke Toda
On the Power of Generalized MOD-Classes

92-12 V. Arvind, J. Köbler, M. Mundhenk
Reliable Reductions, High Sets and Low Sets

92-13 Alfons Geser
On a monotonic semantic path ordering

92-14* Joost Engelfriet, Heiko Vogler
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 Alfred Lupper, Konrad Froitzheim
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

The COCOON Object Model

93-03 Thomas Thierauf, Seinosuke Toda, Osamu Watanabe
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 Jin-Yi Cai, Frederic Green, Thomas Thierauf
On the Correlation of Symmetric Functions

93-05 K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam
A Conceptual Approach to an Open Hospital Information System

93-06 Klaus Gaßner
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 Ullrich Keßler, Peter Dadam
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 Michael Schmitt
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 Armin Kühnemann, Heiko Vogler
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 Harry Buhrman, Jim Kadin, Thomas Thierauf
On Functions Computable with Nonadaptive Queries to NP

94-04 Heinz Faßbender, Heiko Vogler, Andrea Wedel
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers
On Helping and Interactive Proof Systems

Incorporating record subtyping into a relational data model

A Classification of Multi-Database Languages

Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

Construction and Deduction Methods for the Formal Development of Software

Formalisierung schematischer Algorithmen

New Collapse Consequences of NP Having Small Circuits

On Average Polynomial Time

Towards Average-Case Complexity Analysis of NP Optimization Problems

Linking Reactive Software to the X-Window System

Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

Verteilte Unix-Betriebssysteme

Again on Recognition and Parsing of Context-Free Grammars: Two Exercises in Transformational Programming

Transformational Development of Data-Parallel Algorithms: an Example

On the Largest Common Subgraph Problem

Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

The Complexity of Generating and Checking Proofs of Membership

Structural Average Case Complexity

Architecture Independent Massive Parallelization of Divide-And-Conquer Algorithms
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>95-06</td>
<td>Christoph Karg, Rainer Schuler</td>
<td>Structure in Average Case Complexity</td>
</tr>
<tr>
<td>95-07</td>
<td>P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe</td>
<td>ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger kooperierender Assistenzsysteme in klinischen Anwendungsumgebungen</td>
</tr>
<tr>
<td>95-08</td>
<td>Jürgen Kehrer, Peter Schulthess</td>
<td>Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik</td>
</tr>
<tr>
<td>95-09</td>
<td>Hans-Jörg Burtschick, Wolfgang Lindner</td>
<td>On Sets Turing Reducible to P-Selective Sets</td>
</tr>
<tr>
<td>95-10</td>
<td>Boris Hartmann</td>
<td>Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen Netzen am Beispiel Truck Backer-Upper</td>
</tr>
<tr>
<td>95-12</td>
<td>Klaus Achatz, Wolfram Schulte</td>
<td>Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists</td>
</tr>
<tr>
<td>95-13</td>
<td>Andrea Mößle, Heiko Vogler</td>
<td>Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes</td>
</tr>
<tr>
<td>96-01</td>
<td>Ercüment Canver, Jan-Tecker Gayen, Adam Moik</td>
<td>Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche mit VSE</td>
</tr>
<tr>
<td>96-02</td>
<td>Bernhard Nebel</td>
<td>Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class</td>
</tr>
<tr>
<td>96-03</td>
<td>Ton Vullinghs, Wolfram Schulte, Thilo Schwinn</td>
<td>An Introduction to TkGofer</td>
</tr>
<tr>
<td>96-04</td>
<td>Thomas Beuter, Peter Dadam</td>
<td>Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am Beispiel der Domäne Concurrent-Engineering</td>
</tr>
<tr>
<td>96-05</td>
<td>Gerhard Schellhorn, Wolfgang Ahrendt</td>
<td>Verification of a Prolog Compiler - First Steps with KIV</td>
</tr>
<tr>
<td>96-06</td>
<td>Manindra Agrawal, Thomas Thierauf</td>
<td>Satisfiability Problems</td>
</tr>
<tr>
<td>96-07</td>
<td>Vikraman Arvind, Jacobo Torán</td>
<td>A nonadaptive NC Checker for Permutation Group Intersection</td>
</tr>
<tr>
<td>96-08</td>
<td>David Cyrluk, Oliver Möller, Harald Rueß</td>
<td>An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with Composition and Extraction</td>
</tr>
<tr>
<td>96-09</td>
<td>Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte</td>
<td>Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT-Ansätzen</td>
</tr>
</tbody>
</table>
96-10  Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Formalizing Fixed-Point Theory in PVS

96-11  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Mechanized Semantics of Simple Imperative Programming Constructs

96-12  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Generic Compilation Schemes for Simple Programming Constructs

96-13  Klaus Achatz, Helmuth Partsch
From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants

97-01  Jochen Messner
Pattern Matching in Trace Monoids

97-02  Wolfgang Lindner, Rainer Schuler
A Small Span Theorem within P

97-03  Thomas Bauer, Peter Dadam
A Distributed Execution Environment for Large-Scale Workflow Management Systems with Subnets and Server Migration

97-04  Christian Heinlein, Peter Dadam
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow Dependencies

97-05  Vikraman Arvind, Johannes Köbler
On Pseudorandomness and Resource-Bounded Measure

97-06  Gerhard Partsch
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den digitalen Mobilfunkstandard DECT

97-07  Manfred Reichert, Peter Dadam
ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08  Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler
The Project NoName - A functional programming language with its development environment

97-09  Christian Heinlein
Grundlagen von Interaktionsausdrücken

97-10  Christian Heinlein
Graphische Repräsentation von Interaktionsausdrücken

97-11  Christian Heinlein
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12  Gerhard Schellhorn, Wolfgang Reif
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem Provers
97-13 Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

97-14 Wolfgang Reif, Gerhard Schellhorn
Theorem Proving in Large Theories

97-15 Thomas Wennekers
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

97-16 Peter Dadam, Klaus Kuhn, Manfred Reichert
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

97-17 Mohammad Ali Livani, Jörg Kaiser
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

98-01 Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

98-02 Thomas Bauer, Peter Dadam
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

98-03 Marko Luther, Martin Strecker
A guided tour through Typelab

98-04 Heiko Neumann, Luiz Pessoa
Visual Filling-in and Surface Property Reconstruction

98-05 Ercüment Canver
Formal Verification of a Coordinated Atomic Action Based Design

98-06 Andreas Küchler
On the Correspondence between Neural Folding Architectures and Tree Automata

98-07 Heiko Neumann, Thorsten Hansen, Luiz Pessoa
Interaction of ON and OFF Pathways for Visual Contrast Measurement

98-08 Thomas Wennekers
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

98-09 Thomas Bauer, Peter Dadam
Variable Migration von Workflows in ADEPT

98-10 Heiko Neumann, Wolfgang Sepp
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

98-11 Frank Houdek, Dietmar Ernst, Thilo Schwinn
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment
Gerhard Schellhorn
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

Gerhard Schellhorn, Wolfgang Reif
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

Mohammad Ali Livani
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

Mohammad Ali Livani, Jörg Kaiser
Predictable Atomic Multicast in the Controller Area Network (CAN)

Susanne Boll, Wolfgang Klas, Utz Westermann
A Comparison of Multimedia Document Models Concerning Advanced Requirements

Thomas Bauer, Peter Dadam
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

Uwe Schöning
On the Complexity of Constraint Satisfaction

Ercument Canver
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

Johannes Köbler, Wolfgang Lindner, Rainer Schuler
Derandomizing RP if Boolean Circuits are not Learnable

Utz Westermann, Wolfgang Klas
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

Peter Dadam, Manfred Reichert

Vikraman Arvind, Johannes Köbler
Graph Isomorphism is Low for ZPPNP and other Lowness results

Thomas Bauer, Peter Dadam
Efficient Distributed Workflow Management Based on Variable Server Assignments

Thomas Bauer, Peter Dadam
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

Gregory Baratoff, Christian Toepfer, Heiko Neumann
Combined space-variant maps for optical flow based navigation

Wolfgang Gehring
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen
2000-05  Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06  Wolfgang Reif, Gerhard Schellhorn, Andreas Thums
Fehlersuche in Formalen Spezifikationen

2000-07  Gerhard Schellhorn, Wolfgang Reif (eds.)

2000-08  Thomas Bauer, Manfred Reichert, Peter Dadam
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09  Thomas Bauer, Peter Dadam
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10  Thomas Bauer, Manfred Reichert, Peter Dadam
Adaptives und verteiltes Workflow-Management

2000-11  Christian Heinlein
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01  Hubert Hug, Rainer Schuler
DNA-based parallel computation of simple arithmetic

2001-02  Friedhelm Schwenker, Hans A. Kestler, Günther Palm
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03  Hans A. Kestler, Friedhelm Schwenker, Günther Palm
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04  Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01  Stefanie Rinderle, Manfred Reichert, Peter Dadam
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02  Walter Guttmann
Deriving an Applicative Heapsort Algorithm

2002-03  Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01  Manfred Reichert, Stefanie Rinderle, Peter Dadam
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02  Stefanie Rinderle, Manfred Reichert, Peter Dadam
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03  Christian Heinlein
Safely Extending Procedure Types to Allow Nested Procedures as Values
2003-04  Stefanie Rinderle, Manfred Reichert, Peter Dadam  
On Dealing With Semantically Conflicting Business Process Changes.  

2003-05  Christian Heinlein  
Dynamic Class Methods in Java  

2003-06  Christian Heinlein  
Vertical, Horizontal, and Behavioural Extensibility of Software Systems  

2003-07  Christian Heinlein  
Safely Extending Procedure Types to Allow Nested Procedures as Values  
(Corrected Version)  

2003-08  Changeling Liu, Jörg Kaiser  
Survey of Mobile Ad Hoc Network Routing Protocols  

2004-01  Thom Frühwirth, Marc Meister (eds.)  
First Workshop on Constraint Handling Rules  

2004-02  Christian Heinlein  
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined Operator Symbols and Control Structures  

2004-03  Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)  
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence  

2005-01  Armin Wolf, Thom Frühwirth, Marc Meister (eds.)  
19th Workshop on (Constraint) Logic Programming  

2005-02  Wolfgang Lindner (Hg.), Universität Ulm, Christopher Wolf (Hg.) KU Leuven  
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm  

2005-03  Walter Guttmann, Markus Maucher  
Constrained Ordering  

2006-01  Stefan Sarstedt  
Model-Driven Development with ACTIVECHARTS, Tutorial  

2006-02  Alexander Raschke, Ramin Tavakoli Kolagari  
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten Systemen  

2006-03  Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari  
Eine qualitative Untersuchung zur Produktlinien-Integration über Organisationsgrenzen hinweg  

2006-04  Thorsten Liebig  
Reasoning with OWL - System Support and Insights –  

2008-01  H.A. Kestler, J. Messner, A. Müller, R. Schuler  
On the complexity of intersecting multiple circles for graphical display
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008-03</td>
<td>Frank Raiser</td>
<td>Semi-Automatic Generation of CHR Solvers from Global Constraint Automata</td>
</tr>
<tr>
<td>2008-04</td>
<td>Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander</td>
<td>Entscheidungsdocumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse</td>
</tr>
<tr>
<td>2008-05</td>
<td>Markus Kalb, Claudia Dittrich, Peter Dadam</td>
<td>Support of Relationships Among Moving Objects on Networks</td>
</tr>
<tr>
<td>2008-06</td>
<td>Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)</td>
<td>WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke</td>
</tr>
<tr>
<td>2008-07</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>An empirical assessment of local and population based search methods with different degrees of pseudorandomness</td>
</tr>
<tr>
<td>2008-08</td>
<td>Henning Wunderlich</td>
<td>Covers have structure</td>
</tr>
<tr>
<td>2008-09</td>
<td>Karl-Heinz Niggl, Henning Wunderlich</td>
<td>Implicit characterization of FPTIME and NC revisited</td>
</tr>
<tr>
<td>2008-10</td>
<td>Henning Wunderlich</td>
<td>On span-P^{cc} and related classes in structural communication complexity</td>
</tr>
<tr>
<td>2008-11</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>On the different notions of pseudorandomness</td>
</tr>
<tr>
<td>2008-12</td>
<td>Henning Wunderlich</td>
<td>On Toda’s Theorem in structural communication complexity</td>
</tr>
<tr>
<td>2008-13</td>
<td>Manfred Reichert, Peter Dadam</td>
<td>Realizing Adaptive Process-aware Information Systems with ADEPT2</td>
</tr>
<tr>
<td>2009-01</td>
<td>Peter Dadam, Manfred Reichert</td>
<td>The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support Challenges and Achievements</td>
</tr>
<tr>
<td>2009-03</td>
<td>Alena Hallerbach, Thomas Bauer, Manfred Reichert</td>
<td></td>
</tr>
</tbody>
</table>
Correct Configuration of Process Variants in Provop

2009-04 Martin Bader
On Reversal and Transposition Medians

2009-05 Barbara Weber, Andreas Lanz, Manfred Reichert
Time Patterns for Process-aware Information Systems: A Pattern-based Analysis

2009-06 Stefanie Rinderle-Ma, Manfred Reichert
Adjustment Strategies for Non-Compliant Process Instances

Statistical Computing 2009 – Abstracts der 41. Arbeitstagung

2009-08 Ulrich Kreher, Manfred Reichert, Stefanie Rinderle-Ma, Peter Dadam
Effiziente Repräsentation von Vorlagen- und Instanzdaten in Prozess-Management-Systemen

2009-09 Dammertz, Holger, Alexander Keller, Hendrik P.A. Lensch
Progressive Point-Light-Based Global Illumination

2009-10 Dao Zhou, Christoph Müssel, Ludwig Lausser, Martin Hopfensitz, Michael Kühl, Hans A. Kestler
Boolean networks for modeling and analysis of gene regulation

2009-11 J. Hanika, H.P.A. Lensch, A. Keller
Two-Level Ray Tracing with Recordering for Highly Complex Scenes

2009-12 Stephan Buchwald, Thomas Bauer, Manfred Reichert
Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen

2010-01 Hariolf Beth, Frank Raiser, Thom Frühwirth
A Complete and Terminating Execution Model for Constraint Handling Rules

2010-02 Ulrich Kreher, Manfred Reichert
Speichereffiziente Repräsentation instanzspezifischer Änderungen in Prozess-Management-Systemen

2010-03 Patrick Frey
Case Study: Engine Control Application

2010-04 Matthias Lohrmann und Manfred Reichert
Basic Considerations on Business Process Quality

2010-05 HA Kestler, H Binder, B Lausen, H-P Klenk, M Schmid, F Leisch (eds):
Statistical Computing 2010 - Abstracts der 42. Arbeitstagung

2010-06 Vera Künzle, Barbara Weber, Manfred Reichert
Object-aware Business Processes: Properties, Requirements, Existing Approaches
<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-01</td>
<td>Flexibilisierung Service-orientierter Architekturen</td>
<td>Stephan Buchwald, Thomas Bauer, Manfred Reichert</td>
</tr>
<tr>
<td>2011-02</td>
<td>Edge-Optimized À-Trous Wavelets for Local Contrast Enhancement with Robust Denoising</td>
<td>Johannes Hanika, Holger Dammertz, Hendrik Lensch</td>
</tr>
<tr>
<td>2011-03</td>
<td>Datenflussvarianten in Prozessmodellen: Szenarien, Herausforderungen, Ansätze</td>
<td>Stefanie Kaiser, Manfred Reichert</td>
</tr>
<tr>
<td>2011-05</td>
<td>PHILharmonicFlows: Research and Design Methodology</td>
<td>Vera Künzle, Manfred Reichert</td>
</tr>
<tr>
<td>2011-06</td>
<td>Ensuring Business Process Compliance Along the Process Life Cycle</td>
<td>David Knuplesch, Manfred Reichert</td>
</tr>
</tbody>
</table>