Information Systems Frontiers, Springer manuscript No.
(will be inserted by the editor)

On Enabling Integrated Process Compliance with Semantic Constraints in Process Management Systems

Requirements, Challenges, Solutions

Linh Thao Ly · Stefanie Rinderle-Ma · Kevin Göser · Peter Dadam

the date of receipt and acceptance should be inserted later

Abstract Key to broad use of process management systems (PrMS) in practice is their ability to foster and ease the implementation, execution, monitoring, and adaptation of business processes while still being able to ensure robust and error-free process enactment. To meet these demands a variety of mechanisms has been developed to prevent errors at the *structural* level (e.g., deadlocks). In many application domains, however, processes often have to comply with business level rules and policies (i.e., *semantic constraints*) as well. Hence, to ensure error-free executions at the semantic level, PrMS need certain control mechanisms for validating and ensuring the compliance with semantic constraints. In this paper, we discuss fundamental requirements for a comprehensive support of semantic constraints in PrMS. Moreover, we provide a survey on existing approaches and discuss to what extent they are able to meet the requirements and which challenges still have to be tackled. In order to tackle the particular challenge of providing integrated compliance support over the process lifecycle, we introduce the SeaFlows framework. The framework introduces a behavioural level view on processes which serves a conceptual process representation for constraint specification approaches. Further, it provides general compliance criteria for static compliance validation but also for dealing with process changes. Altogether, the SeaFlows framework can serve as formal basis for realizing integrated support of semantic constraints in PrMS.

Keywords Adaptive process management systems · Semantic constraints · Process verification · Compliance validation

This work was done within the research project "SeaFlows: Semantic Constraints in Process Management Systems", which is funded by the German Research Foundation (DFG). http://www.uni-ulm.de/en/in/iui-dbis/research/projects/seaflows.html

Linh Thao Ly
Ulm University
Institute of Databases and Information Systems
Tel.: +49-731-5024194
Fax: +49-731-5024134
E-mail: thao.ly@uni-ulm.de

Stefanie Rinderle-Ma
Ulm University
Institute of Databases and Information Systems
E-mail: stefanie.rinderle@uni-ulm.de

Kevin Göser
AristaFlow GmbH
E-mail: kevin.goeser@aristaflow.com

Peter Dadam
Ulm University
Institute of Databases and Information Systems
E-mail: peter.dadam@uni-ulm.de

1 Introduction

Due to continuously changing market conditions, organisations are forced to frequently adapt their business strategies in order to stay competitive [4,13,17, 59,71]. Hence, there is a strong demand for process-aware information systems facilitating fast implementation and deployment of business processes and allowing for their flexible adaptation. Process management systems (PrMS) are supposed to fulfil these demands and therefore are gaining increasing importance. Key to the application of PrMS technology in practice is their ability to allow for fast and flexible realization of business processes on the one hand, while still being able to ensure error-free process executions on the other hand [14].

So far, research emphasis has been put on avoiding errors at the *structural level* [2,47,58,59]. For example, by checking whether a process model contains deadlocks or incorrect data links at design time, a PrMS can guarantee for the absence of *structural* errors during process execution. Even if processes have to be adapted in order to handle exceptional situations (e.g., by inserting additional activities, moving activities to other positions, or by performing even more complex change operations [57]), these checks can be used for ensuring the *structural correctness* of process changes [58]. Respective control mechanisms for process modelling and execution make PrMS an appealing development and execution environment for business processes.

1.1 Problem Statement

Supporting solely checks at the structural level of processes, however, is not sufficient to ensure their error-free execution. In many application domains, processes are subject to business level rules and policies stemming from domain specific requirements (e.g., standardisation, legal regulations) [47,60]. In the clinical domain, for example, clinical guidelines and pathways [38,53] can be considered as examples thereof. To clearly distinguish between structural constraints and business level constraints stemming from compliance requirements, we refer to the latter as *semantic constraints*. Semantic constraints may express various dependencies such as ordering and temporal relations between activities, incompatibilities, and existence dependencies. Hence, the semantic constraints addressed in this paper can be considered a subset of *business rules* [64,66]. As examples consider the constraints we have collected from different domains (particularly healthcare, banking, and product release management) in Tab. 1.

The feasibility of manually assessing whether or not processes comply with imposed semantic constraints is very limited. This especially becomes true when considering complex processes involving hundreds of tasks and related data flows [12]. Validating and ensuring compliance with semantic constraints will get even more challenging if dynamic process changes are allowed during process execution [57]. Hence, there is an evident demand for control mechanisms enabling PrMS to support the validation and enforcement of semantic constraints at the system level.

Compliance validation has been addressed from various perspectives (e.g., business process compliance [60, 51,22,23,40,36], and compliance of cross-organisational workflows with business contracts [28,32], compliance of workflow transactions with predefined dependencies [9, 63]). Most existing approaches either follow the paradigm of compliance validation at process model level (design time) or compliance monitoring at process instance level (runtime). However, taking the process lifecycle [49, 68] into account, we believe that PrMS have to provide more comprehensive support of semantic constraints. In particular, PrMS must be able to ensure compliance over the complete process lifecycle (*life time compliance*).

Table 1 Examples of semantic constraints

	Semantic constraints in natural language
c_1	A patient should not be administered the drugs Aspirin and Marcumar within 5 days due to possible unwanted interactions.
c_2	For patients older than 75 an additional tolerance test prior to the examination is required due to an increased risk.
c_3	No endoscopic examinations shall take place within one week after radiological examinations using non-water-soluble contrast agents.
c_4	A radiological examination of an inpatient has to be followed by a ward round (whereas ambulatory radiological examinations do not require subsequent ward rounds).
c_5	The patient has to be informed prior to invasive procedures.
c_6	The approval of loan applications with a loan amount greater than 60.000 € has to be checked by the manager of the loan department before granting.
c_7	The assembly always has to be followed by a technical acceptance and a test run. Additionally, no further adaptations must take place between the technical acceptance and the test run.

1.2 Contribution

In previous work [43], we introduced a basic set of semantic constraints (i.e., binary exclusion and dependency constraints) expressing interdependencies between process activities. Furthermore, we presented optimization techniques for validating processes against these constraints by restricting the set of relevant constraints based on the semantics of the applied change operations [44]. This approach provides mechanisms for ensuring compliance not only at design time, but also at runtime. In the course of further studies, however, we noticed that many application scenarios require even

more expressive constraints (cf. Tab. 1). This poses additional requirements on their integrated support in PrMS.

In this paper, we discuss the challenge of supporting semantic constraints in PrMS from a holistic point of view. For this purpose, we first provide a detailed discussion on fundamental requirements for supporting semantic constraints in a comprehensive manner. Furthermore, we discuss to what extent existing approaches are able to meet these requirements and show which challenges still have to be tackled.
In the second part of the paper, we address the particular challenge of enabling life time compliance. We advocate that life time compliance can only be achieved by providing an overall framework with adequate mechanisms for supporting compliance in each phase of the process lifecycle. To tackle this, we propose a general framework developed in the SeaFlows project. The SeaFlows framework introduces a trace-based behavioural level view on processes which serves a conceptual process representation for constraint specification approaches. This conceptual representation is a suitable underlying logical model for compliance validation throughout the process lifecycle. Based on the behavioural level view, the framework further provides general compliance criteria for assessing the compliance of processes with semantic constraints in all phases of the process lifecycle. Furthermore, it provides criteria for assessing the effects of process changes regarding compliance with semantic constraints. Altogether, the SeaFlows framework can serve as formal foundation for realizing integrated support of semantic constraints throughout the process lifecycle. The SeaFlows prototype implementing ideas of the framework shows directions of our future research.

The remainder of this paper is organised as follows. Fundamental requirements are elaborated in Sect. 2. In Sect. 3, state of the art is discussed with regard to these requirements. Our vision of enabling life time compliance is presented in Sect. 4. Sect. 5 introduces the formal framework providing the foundations for realizing this vision. The basic ideas are described in Sect. 5.1. The behavioural level view on processes is explained in Sect. 5.2. Based on this, formal compliance criteria are introduced in Sect. 6. Sect. 7 provides considerations on applying the formal framework and presents the SeaFlows prototype. Finally, a summary and an outlook on future research are provided in Sect. 8.

2 Fundamental Requirements for Supporting Semantic Constraints

For supporting semantic constraints in PrMS, existing PrMS concepts have to be complemented by mechanisms for specifying respective constraints and for assigning them to processes. Furthermore, mechanisms for validating and ensuring the compliance of processes with the semantic constraints have to be provided. From case studies (particularly of clinical processes, e.g. [37]) we derived fundamental requirements which have to be considered by a comprehensive approach and which are discussed in the following.

2.1 Specifying and Integrating Constraints

2.1.1 A Formal Language for Constraint Specification (Req. 1)

When designing or choosing a constraint specification language we have to deal with several trade-offs. On the one hand, a constraint specification language has to provide the expressiveness necessary to model real world semantic constraints. On the other hand, the expressiveness must not be achieved at the expense of validation and analysis costs. Especially large constraint sets demand for mechanisms for formal analysis to ensure their consistency (i.e., no contradicting constraints). This, in turn, demands for a constraint specification language which has a sound formal foundation [60]. In addition, the complexity of the specification language must neither become an obstacle for constraint specification nor for the validation of processes against constraints. Thus, the main challenge is to find an appropriate balance between expressiveness, formal foundation, and efficient analysis.

2.1.2 Constraint Organisation (Req. 2)

Though there exists semantic constraints only relevant for one particular process (i.e., process-specific constraints), many constraints are relevant to multiple processes. An example thereof is the relation between the endoscopic examinations and radiological examinations with non-water-soluble contrast agents (cf. Tab. 1). Hence, an appropriate way of organising semantic constraints (e.g., in a constraint repository [43] or a directory [60]) has to be provided in order to support the process-spanning specification and (re)use of semantic constraints.

Similar to processes, semantic constraints may change and thus, are subject to an evolution process. This is particularly true for third party constraints.

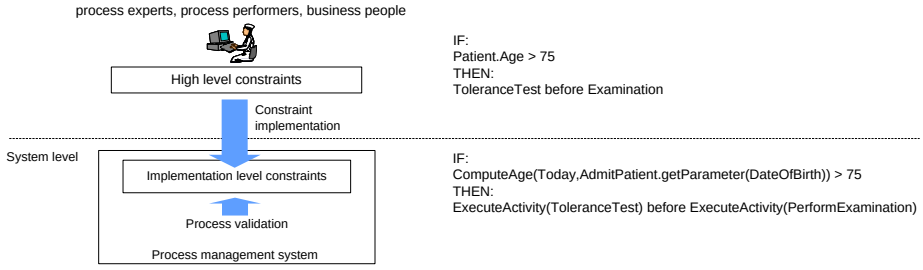


Fig. 1 Support of two abstraction levels for semantic constraints

Clinical guidelines, for example, may change due to new evidences in healthcare [53,38]. Since the lifecycle of constraints and the lifecycle of processes do not necessarily coincide, mechanisms for versioning and propagating constraint changes to relevant processes have to be provided to support constraint evolution.

2.1.3 Views on Semantic Constraints at Different Abstraction Levels (Req. 3)

We identified two oppositional requirements regarding the abstraction level of employed in semantic constraints (cf. Fig. 1). Since semantic constraints need to be understood, managed, and specified by domain experts (e.g., process performers, business people) [36], a high level view on semantic constraints abstracting from implementation details has to be provided. Further, constraints (or what they refer to) may be implemented in various ways in a particular process. Thus, specifying semantic constraints at implementation level (i.e., hard-wiring constraints) would restrict the (re)use of the constraints (cf. Req. 2). This will be particularly important if process implementations may be replaced or changed over time. In this case, a constraint not abstracting from process implementation details would have to be revised and adapted to match the new process implementation even though its semantics has not changed. For example, consider the constraint c_2 from Tab. 1. For the semantics of c_2, it is irrelevant how the age of a patient will be finally determined in a particular process implementation; i.e., whether there will be a designated context data element in the process corresponding to the patient's age or whether the latter will have to be computed from the patient's date of birth. This example indicates the demand for an abstract view on semantic constraints. However, implementation level constraints are indispensable for process validation. For example, for validating whether or not an additional tolerance test for a patient is required in a particular process (according to c_2), the PrMS has to know exactly how to determine the patient's age in this process. In summary, both a high level (i.e., conceptual) view on constraints focussing on their semantics and an implementation level view for constraint evaluation are essential. This, in turn, also raises the requirement of providing adequate mechanisms for mapping these views onto each other.

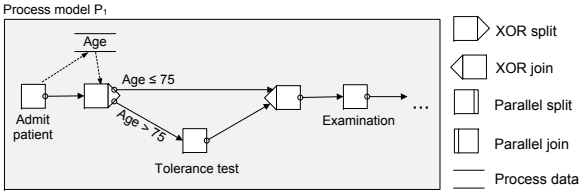


Fig. 2 A compliant process model

2.2 Ensuring Compliance

2.2.1 Support of Life Time Compliance (Req. 4)

Taking the process lifecycle [49,68] into account, we identify four scenarios for semantic process validation:

Compliance Validation at Design Time (Req. 4.1)
Generally, it is desirable to ensure compliance of a process with semantic constraints already at the modelling level (*compliance by design* [60]). A process model will be denoted as compliant with a set of semantic constraints, if it only allows for the execution of process instances not violating these constraints. Thus, by ensuring compliance at process model level, it is ensured that corresponding process instances are compliant as well. As example consider process model P_1 in Fig. 2. P_1 only allows for executing process instances which are compliant with constraint c_2 (i.e., P_1 is compliant with c_2). For enforcing compliance at process model level, mechanisms have to be provided in order to validate process models.

Compliance Validation at Runtime (Req. 4.2)
Being able to validate process models against constraints is essential to enable compliance by design. However, it is not always feasible to enforce compliance with all constraints imposed on a process model at design time (i.e.,

by "compiling" the constraints into the process model). As example consider a process which has to comply with a large set of clinical guidelines with directives on what actions to take in exceptional cases (e.g., in case of a particular allergy). Enforcing the compliance with all these *context-related* constraints at the process model level might lead to a highly complex process model since each possible case described in a constraint has to be accounted for in the process model (e.g., by inserting corresponding conditional branches into the process model). Hence, depending on the nature of the constraints (e.g., how often an allergy occurs), it could be more practicable to postpone the enforcement of compliance with these constraints until runtime.
Semantic constraints involving unexpected events (e.g., if a patient's leukocyte count suddenly falls below a threshold, a drug for raising the leukocyte count will have to be administered the same day) also require adequate mechanisms for runtime monitoring and validation. Such events cannot be anticipated and, thus, the constraint cannot be enforced properly at model level without overcomplicating the process model.

Validation of Process Changes (Req. 4.3)
Compliance validation also becomes necessary when process instances have to be modified during runtime in order to deal with exceptional situations [44]. Particularly if a process instance is frequently modified in an ad hoc manner by various agents with only restricted view on the process, conflicts between process changes and semantic constraints may occur. An example of how a process change can lead to a semantic inconsistency is given in Fig. 3. Instance I_1 is modified by deleting the tolerance test (e.g., due to lack of time). This deletion, however, violates constraint c_2 since no tolerance test is carried out though the patient is older than 75. To allow for flexible process execution and to avoid that flexibility leads to semantic inconsistencies in processes, PrMS have to provide mechanisms to validate the compliance of process changes. For this purpose, semantic constraints potentially affected by the process change have to be identified and evaluated. Note that it might become necessary to *reevaluate* semantic constraints with regard to the requested process changes. Moreover, mechanisms to enforce compliance, for example, by warning the user of conflicting changes become essential as well. Since runtime validation often involves interaction with end users, efficient runtime checks are needed.

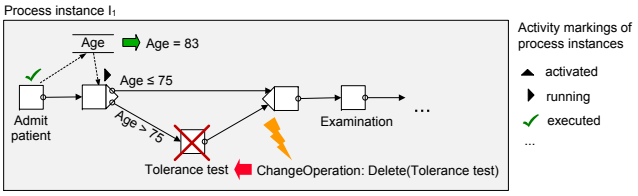


Fig. 3 A process instance change leading to semantic inconsistencies

Compliance Validation for Process Evolution (Req. 4.4)
When a process model is adapted (e.g., due to process optimization) it is often desirable that instances being executed according to the old model version also benefit from these changes [58,59] (*change propagation*).

Checking whether process instances can be migrated to the new process model (i.e., whether changes to the model can be propagated to running instances) with regard to semantic constraints is not a trivial question. Firstly, the number of instances to be checked can become very large as there might be thousands of running process instances of a process model in a process-aware information system [57]. Secondly, process instances may have been individually modified. For example, instance I_2 in Fig. 4 has been modified by inserting a CT examination. Hence, changes at the model level may be conflicting with changes at the instance level [44]. For example, the propagation of the insertion of the endosonography in P_2 to I_2 could be conflicting with regard to c_3. To deal with such scenarios adequate checks supporting the propagation of process model changes to process instances in terms of compliance with semantic constraints are essential.

2.2.2 Support Process-spanning Scenarios (Req. 5)

Processes may be semantically related to each other. For example, it can become necessary to split an overall process into several physical processes in a PrMS (e.g., complex treatment processes involving different organisational departments). However, business processes which are as such independent from each other may also be interrelated due to being carried out for the same artefacts (e.g., several instances of treatment processes for one patient or several insurance claims for the same client). This leads to the situation that the scope of semantic constraints may reach across multiple processes. Consider, for example, the process instances depicted in Fig. 5. Two examinations have been initiated for patient Smith. These instances cause a potential conflict with regard to constraint c_3 from Tab. 1 since an endoscopy is about to be carried out after a radiological examination (i.e., the CT examination). Hence, the situation described above shows the necessity to ensure the compliance with semantic constraints across multiple processes. This demands for mechanisms which al-

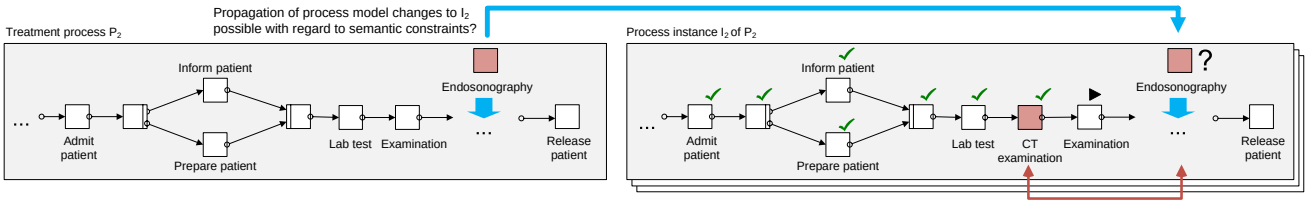


Fig. 4 Process evolution and resulting requirements for semantic constraint support

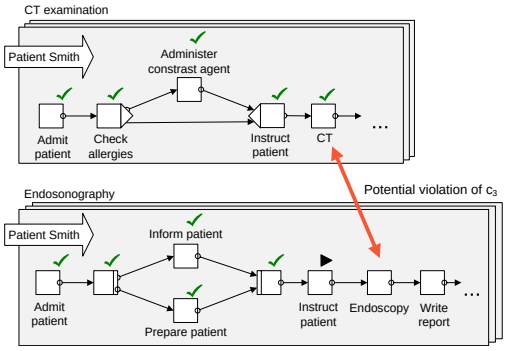


Fig. 5 Two process instances for the same patient causing a potential violation

low to identify the relevant processes in the first place (e.g., the treatment processes for patient Smith). Furthermore, technical solutions are required in order to validate and ensure the compliance of constraints across process boundaries.

2.2.3 Providing Intelligible Feedback (Req. 6)

Semantic constraints are supposed to govern process executions to ensure a semantically consistent behaviour. Thus, many interactions with users may occur (e.g., in case of constraint violations). Therefore, intelligible feedback is highly important for user acceptance. Especially in case of (potential) constraint violations, helpful feedback providing an error diagnosis is required. In addition, feedback to assist the user in applying adequate conflict avoidance (e.g., abstaining from a process change) and compensation strategies (e.g., inserting a compensatory activity) is essential as well.

2.2.4 Support of Flexible Constraint Handling (Req. 7)

Semantic constraints are often not stringent. For example, physicians may deviate from predefined recommendations in clinical guidelines [38]. Hence, constraint-awareness of PrMS must not conflict with the need for flexible processes. In particular, constraint violations need not necessarily be an error but could also be intended. Prohibiting constraint overriding in such cases could annoy users and might even cause them to bypass the system. Therefore, it must be possible to override semantic constraints during process execution depending on the constraint and the situation. This, in turn, necessitates the introduction of enforcement levels and corresponding classification of constraints in the first place. Further, to define enforcement strategies (e.g., "Only physicians are permitted to override this constraint"), it must be possible to relate constraints to organisational structures (e.g., roles). Here we can benefit from the research on business rules and particularly on business rules classification and properties [50]. Since the semantic constraints addressed in this paper essentially constitute a subclass of business rules (i.e., *constraints* or *action assertions*) [64], these results can be adopted for constraint classification as well. For example, enforcement levels (e.g., *strict*, *deferred*, *override*, and *guideline* [65]) and modalities [54] stemming from modal logic [11] from the business rules world can be adopted to enable the definition of advanced strategies for flexible constraint handling in PrMS.

At the technical level, overriding of constraints at the process instance level also unfolds further challenges when dealing with process evolution (cf. Req. 4.4). In particular, process models and process instances not necessarily obey the same constraints due to constraint overriding for individual process instances. This possibly affects the propagation of process model changes to running process instances.

2.2.5 Support of Traceability (Req. 8)

Since traceability is highly important in general, the results of semantic process checks have to be documented. Then, it becomes possible to reconstruct past compliance checks and corresponding results. This is particularly necessary when it comes to constraint violations or constraint overriding. In this case, it has to be recorded who initiated the overriding and for what reasons. In the clinical domain, for example, such information is needed to establish interdependencies between the adherence to guidelines and the process outcome [38,55].

3 State of the Art

In PrMS research, emphasis has been put on developing techniques for ensuring *structural correctness* of processes [2,58,59,71]. These techniques address the validation of both control and data flow. Due to recent compliance requirements (e.g., Sarbanes-Oxley Act [45] and EuroSOX [1]) *business process compliance* has dramatically gained importance for organisations and also in BPM research [60]. Enabling PrMS to support the compliance of processes with imposed semantic constraints can be regarded as one step towards the implementation of business process compliance [36].

There are many approaches addressing the particular issue of constraint acquisition, specification, and formalisation. Many of these stem from business rules research [66,64] and standardisation (e.g., OMG SBVR [54]). The detailed discussion of constraint specification and formalisation approaches could easily constitute a separate paper. For this reason such a discussion is beyond the scope of this paper. The interested reader is referred to [31,50,54] for detailed discussions.

In this section, we discuss approaches from PrMS research as well as related research areas (e.g., SOA and web services) which focus on ensuring the compliance of processes with constraints in a broader sense. Existing approaches are first elaborated with regard to the validation scenario they focus on (cf. Req. 4). A discussion of the other requirements follows in Sect. 3.5.

3.1 A Priori Compliance – Design Time Validation

Common to approaches in this category is the basic idea to achieve compliance by validating a process specification (i.e., a process model) against certain constraints already at design time (cf. Req. 4.1). Existing approaches vary in used constraint specification languages, validation techniques, and backgrounds.

In [61], an approach for achieving flexible processes is described which allows for the late modelling of subprocesses. Constraints expressing dependencies between activities are introduced for restricting composition possibilities. Before a subprocess is executed, it is validated against the constraints. This approach particularly allows for balancing between flexibility and control. This is achieved by enabling the definition of process models ranging from fully modelled to mainly constraint-based. In [15], an approach for compliance validation based on concurrent transaction logic (CTR) is introduced. For validating a process model against constraints specified in CTR, the process graph is transformed into a CTR formula. This allows for the application of reasoning techniques for identifying semantic conflicts. Förster et al. [20] present an approach for validating process models against quality constraints. Quality constraints are specified in terms of process patterns in process pattern specification language (PPSL). PPSL patterns, in turn, can be transformed into specifications in linear temporal logic (LTL). Process models are validated against quality constraints by applying model checking techniques. Ghose et al. [22] introduce an approach for auditing BPMN process models for compliance by annotating activities with effects. The latter correspond to propositions in LTL. This enables the application of model checking techniques for validation.

Lu et al. [41] introduce an interesting approach for measuring the compliance distance between a process model and a set of control objectives (comparable to constraints). The latter are specified in formal contract language (FCL) [28]. Compliance is measured by comparing possible execution traces of the process model against ideal and sub-ideal execution traces. In [27], an approach based on annotating activities with their effects represented by logical propositions is introduced. FCL is used to formulate constraints (i.e., FCL rules specifying under which state conditions certain obligations arise). By propagating the effects of activities throughout the process model, obligations sure to arise during process execution can be detected and evaluated. A similar propagation approach is used in [69] in order to approximate compliance checking. This approach aims at detecting states of the process execution in which predefined constraint clauses are violated.

In the context of web service composition and coordination, the question arises whether or not a choreography complies with certain constraints. In [72], Yu et al. introduce an approach for the specification of properties and for the property-based validation of BPEL processes. The properties are based on property patterns [16]. For process validation, a model checking approach is employed. Model checking has also been applied to process validation by several other approaches [21,40]. Foster et al. [19] introduce an approach for validating the interactions of web service compositions against obligation policies specified using message sequence charts. For validation, an approach based on labelled transition systems is employed.

In [28], compliance validation is addressed from the business contract perspective using FCL for specifying contracts. The compliance of a BPMN process with a given contract is validated by transforming the process model into a form similar to the contract notation. This allows for the detection of contract violations in the process model by applying reasoning techniques.

Most a priori validation approaches address dependencies which can be expressed at the activity level.

However, the validation of context-related constraints (e.g., constraints depending on a patient's allergies, a customer's insurance sum, or the outcome of a certain activity) is more complex since the incorporation of context data can lead to state explosion. This, in turn, results in high validation costs. In order to still be able to provide compliance reports for context-related constraints at the process model level, mechanisms for dealing with state explosion have to be integrated.

3.2 Runtime Compliance Validation

A large number of existing approaches focus on runtime compliance validation. The basic idea is to validate compliance by monitoring process-related events during runtime. Early approaches stem from research on rule-based transactions (e.g., [9,63]). Their main focus is to schedule upcoming processing requests (e.g., a commit request) such that predefined constraints (e.g., commit dependencies) are not violated. For specifying and enforcing constraints, logic-based formalisms and techniques (e.g., event algebra [63], temporal logics [9]) are used. In [56], an approach for specifying declarative process models using LTL is presented. For process enactment, the LTL formulas are synthesized into state automatons. In [30], an approach for synchronizing concurrent process instances is introduced. Constraints are specified using an extension of regular expressions. For scheduling process instances according to the constraints an FSM-based coordinator is used.

Monitoring runtime compliance has also been addressed from the business contract perspective (e.g., [32,46,8,23]). In [46], process events are monitored to detect contract violations. In [8], contract clauses are specified in a rule-based form using the notion of *happened*, *expected*, and *not-expected* events. At runtime, events are recorded in a knowledge base which allows for reasoning about contract compliance. [7] employs a similar approach for monitoring the compliance of web service executions with choreographies.

In [52], a semantic mirror (i.e., a knowledge base of process data) is continuously updated according to the current execution status of a process instance. This allows for monitoring the compliance of the instance with constraints specified in terms of event-condition-action rules (ECA). Agrawal et al. [6] also advocate the use of process monitoring for detecting non-compliance. In [29,48], an approach for rule-based automatic instance adaptation is proposed. The ECA rules are specified using active temporal frame logic (an extension of frame logic by temporal notions such as durations). At runtime, upon occurrence of certain events and conditions (such as high blood pressure), the process is automatically adapted according to the rule's action part.

Runtime compliance validation is particularly important for constraints involving runtime context information (cf. Req. 4.2). However, as a limitation most monitoring approaches do not allow for "look-aheads". In particular, possible future process behaviour is treated as unknown. Decisions (e.g., enforcement decisions such as to reject a commit request) can only be made based on execution history so far. This will lead to shortcomings, if the future process behaviour is also relevant to constraints. In the scope of business process management, process models to a certain extend provide a description of the process behaviour. Hence, the process structure described by process models can be exploited at runtime in order to predict and thus avoid non-compliance *in advance*. For this purpose, the integration of an approach for monitoring the state of the process execution into an a priori process validation approach is vital. One challenge for intelligent and efficient monitoring is to trigger the process validation only if necessary (e.g., when new data relevant to a constraint is available). In addition, in order support process users to adequately deal with violations, it is necessary to also enable mechanisms for determining whether violations are still healable at a certain point in time. In this paper, we introduce the basis for dealing with changes of the process execution state. In particular, we discuss the possible effects of state changes on the compliance with semantic constraints. In addition, we introduce a basic notion of healable violations.

3.3 Check Point Metaphor

Business rule management systems (e.g., ILOG JRules [35]) allow for managing and evaluating business rules in a business rule engine (BRE) by employing techniques from knowledge-based systems. In the context of PrMS, a BRE is primarily used for decision making. For this purpose, decision points have to be predefined. At runtime, the rule evaluation will be invoked when reaching a decision point. In the context of service-oriented architecture (SOA), rule evaluation can be implemented as a service encapsulating details specific to the rule engine [39]. In IBM Websphere, for example, hard coded rules and checks are assigned to enforcement points where the rule service will be invoked at runtime [24]. Commercial business process compliance tools (e.g., Bonapart SOX Analyzer [18] and ARIS Solution for GRC [34]) enable the enrichment of processes with risks, controls, and tests. Based on these, workflows for testing the controls can be scheduled. Generally, the check point paradigm complements other

validation scenarios. However, it is not suitable when more flexible checks become necessary (e.g., when processes are adapted) [67].

3.4 A Posteriori Compliance Analysis

In [5], an approach for validating processes a posteriori against constraints is presented. Constraints specified in LTL are verified over process logs. Unfortunately, this approach is not suitable for scenarios in which non-compliance may affect the outcome of a process. However, we consider a posteriori compliance validation an appealing complement to other validation paradigms.

3.5 Further Aspects and Discussion

Though the expressiveness varies, common to most approaches discussed is a certain degree of formal foundation of the specification language (cf. Req. 1).

With SBVR [54] OMG introduces a meta-model which allows for formalising the semantics of business vocabulary and business rules. The formalisation is supposed to enhance the interchange of rules and vocabulary among organisations. The adoption of the differentiation between concept *meaning*, *representation*, and *expression* as proposed by SBVR could be relevant for supporting high level and implementation level specifications and the mappings between the levels (Req. 3). Whether this is already sufficient, however, still has to be investigated. Constraint organisation (Req. 2) and particularly the reuse of existing constraints could also benefit from the formalisation of business vocabulary and rules. The requirement of high level and implementation level constraints (Req. 3) is also addressed by [52]. However, it is not quite clear to what extent it is possible to abstract from implementation details when specifying semantic constraints with this approach.

Monitoring approaches (cf. Sect. 3.2) are potentially able to deal with constraints with process-spanning scope (Req. 5) by nature. This particularly applies to approaches dealing with monitoring compliance with process contracts (e.g. [32,46]). This is because they are designed to deal with cross-organisational scenarios. To evaluate to what extent these approaches are suitable for dealing with semantic constraints, however, a more detailed analysis is required. Many of the other requirements are not within the scope of existing approaches (due to their various backgrounds) and, thus, are not directly addressed (e.g., Req. 2 changes to semantic constraints and Req. 8). In [52], recovery strategies for control violations are proposed (e.g., rollback or ignoring the violation). This is an interesting approach which treats constraint violations the same way as process exceptions. However, these recovery strategies are applied after a constraint has already been violated. Ghose et al. [22] propose interesting compliance patterns which can be applied for conflict resolution. The abstract patterns basically capture commonly occurring violations and provide heuristics to deal with them. In [26], Governatori proposes an interesting strategy for dealing with violations by integrating contrary-to-duty obligations into constraints. Contrary-to-duty obligations represent reparational obligations which arise in case of violations. The notion of contrary-to-duty is particularly helpful in order to distinguish between different ways of satisfying a constraint.
Generally, in case of violations, mechanisms are required to determine which strategies for dealing with violations are applicable (e.g., which users have the rights to override the corresponding constraint). The latter may also depend on the enforcement level of a constraint for a particular process (instance) (cf. Req. 7). For dealing with violations we can also draw further inspiration from other research areas. In agent systems, for example, agents often have to behave in accordance to a predefined protocol. Hence, agent systems research also deals with determining whether or not agents violate constraints and how to deal with violations.

Regarding the requirement of supporting life time compliance (Req. 4), most of the existing approaches either focus on validating process models at design time (Req. 4.1) or on compliance monitoring during runtime (Req. 4.2). In addition, approaches supporting compliance monitoring often do not consider possible future behaviour of processes. The validation of process changes (Req. 4.3) has not been addressed. The same applies to change propagation (Req. 4.4). Hence, although many related approaches offer inspiring solutions for particular facets, to our best knowledge, there is no approach which covers all validation scenarios and allows for integrated compliance support with regard to the process lifecycle. In the second part of this paper, we address Req. 4 and present our ideas on enabling life time compliance. The SeaFlows framework proposed in the remainder of the paper employs the notion of event traces. As discussed, the event notion is also used by some other approaches [63,5,7,8]. In contrast to these, however, we do not propose a particular constraint specification language and corresponding validation mechanisms. In fact, the SeaFlows framework rather uses the event trace notion to provide an integrated conceptual process representation. This is necessary in order to provide an underlying logical model which is suitable for each lifecycle phase. Hence, our approach is also related to approaches dealing with struc-

tural correctness such as [13,17,58,59]. In addition, the SeaFlows framework also introduces general compliance criteria as basis for static but also for dynamic compliance validation.

4 Towards Life Time Compliance – A Vision

For achieving compliance throughout the process lifecycle, adequate mechanisms for supporting and ensuring compliance in each phase are required. Fig. 6 depicts the process lifecycle [49,68] within PrMS enriched with key mechanisms needed for integrated compliance support. In the following, we explain the particular mechanisms envisaged for each lifecycle phase. Due to space limitations, we abstain from presenting our ideas on change propagation (Req. 4.4).

4.1 Design Time: Process Modelling and Composition

4.1.1 Constraint Specification

In order to enable automatic process validation, high level semantic constraints first will have to be transformed into constraints using process artefacts (i.e., implementation level constraints). For this purpose, the PrMS provides an interface for constraint specification based on the process artefacts (e.g., process activities, subprocesses, and process context data) of the domain (cf. Fig. 6 (A)). Depending on how high level constraints are modelled (e.g., formal or informal) the process engineer may transform them by mapping constraint artefacts or by respecifying constraints using the artefacts of the interface. Many approaches for constraint specification have been proposed in literature (e.g., LTL [22,56] and FCL [28]). To identify a suitable constraint specification language, a detailed analysis of relevant semantic constraints is vital. Semantic constraints may be stored in a constraint repository and assigned to categories for facilitating constraint reuse (cf. Fig. 6). To assign semantic constraints to a process, the process engineer can browse the constraint repository for relevant existing constraint sets (e.g., drug interactions) or create new ones.

4.1.2 Process Model Validation

Following Req. 4.1, the PrMS provides mechanisms for process model validation already during design time (cf. Fig. 6 (B)). At this stage, only the process model serves as input for compliance validation. However, many semantic constraints involve runtime information (e.g., context conditions) as well. These context information is not available at design time. Hence, in order to provide the process engineer with helpful validation reports we envisage detailed compliance notions (e.g., conditional violations or definite violation, cf. Fig. 6 (C)). These will allow for more fine-grained compliance prognoses and feedback, which, in turn, might help the process engineer to evaluate and to enhance the process model.

4.2 Runtime: Process Execution and Process Instance Adaptation

It is not always feasible to enforce all semantic constraints at the process model level (cf. Req. 4.2). Hence, it must be also possible to create process instances from a process model which does not enforce all semantic constraints at the structural level. This, in turn, demands for adequate runtime monitoring and validation mechanisms in order to ensure compliance with the constraints not yet enforced. For this purpose, relevant events of the process execution have to be monitored (e.g., availability of relevant context information).
The evaluation of corresponding constraints based on the runtime context then has to be carried out during process execution (cf. Fig. 6 (D)). Our objective is to predict *potential* conflicts (i.e., violations) as early as possible in order to allow for timely application of strategies for averting conflicts. Hence, not only the current execution history of the process instance has to be accounted for, but also the possible future behaviour of the instance (i.e., no mere monitoring).

Compliance checks at design time are less costly than corresponding checks at runtime. Hence, to reduce validation costs, design time and runtime checks should not be performed in an isolated manner. In fact, their interplay has to be supported. For this purpose, it is vital to determine which constraints still have to be monitored and evaluated during execution and which constraints have already been enforced at process model level and thus, do not require costly compliance checks at runtime (cf. Fig. 6 (C)). To further optimize the interplay between design time and runtime validation and particularly to exploit the synergy effects, a detailed analysis of the problem space is required.

Following the requirement for validating process changes (Req. 4.3), corresponding compliance checks have to be integrated into existing process adaptation mechanisms of PrMS (cf. Fig. 6 (E)). Note that a process change may require the reevaluation of semantic constraints which have already been enforced before the change. In order to reduce validation costs, the semantics of the process change to be carried out can

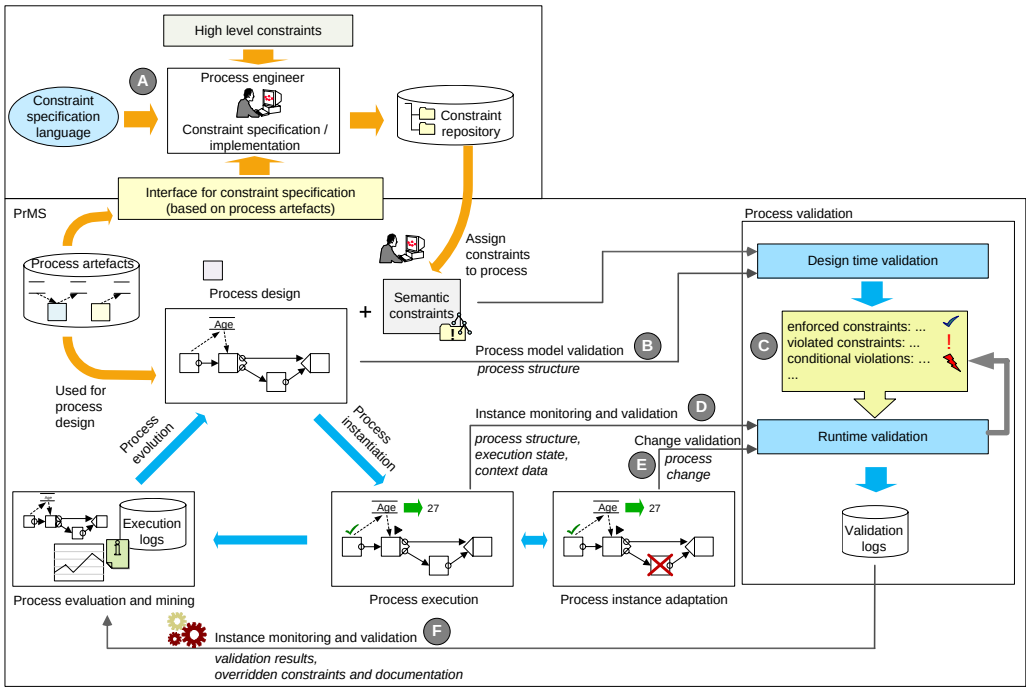


Fig. 6 Key mechanisms for life time compliance

be exploited. In [44], we introduced an approach for (re)evaluating only semantic constraints which might be violated by the particular process change. So far, this approach has been restricted to activity-level constraints and will be extended to handle more expressive constraints.

4.3 Process Evaluation and Mining

Following the requirement for traceability (Req. 8), runtime compliance checks have to be tied with logging mechanisms. This is particularly important when constraints can be overridden during execution (Req. 7). Then, the validation logs can provide meaningful input for process mining (cf. Fig. 6 (F)).

In the context of continuous process learning, a log analysis can help to evaluate and enhance existing semantic constraints (e.g., constraint refinement based on insights on frequently occurring constraint overriding due to a particular reason). This may serve as input to constraint evolution (cf. Req. 2). In addition, a log analysis may also contribute to evaluate the quality of the process by relating process outcome and constraint adherence.

5 A Formal Framework for Integrated Compliance Support

An integrated formal model and corresponding compliance criteria supporting all lifecycle scenarios are prerequisites for enabling integrated compliance support as sketched in Sec. 4. These foundations are provided by the SeaFlows framework introduced in the remainder of this paper. In this paper, we focus on single process scenarios as the basis for compliance validation (i.e., Req. 5 is not addressed). Thus, we do not address synchronisation and scheduling constraints (e.g., as addressed in [30]) which particularly become relevant when multiple processes need to be scheduled. A basic idea how to extend the concepts in this paper in order to deal with multiple process scenarios is sketched in the outlook.

In the following, we first introduce the basic ideas of the SeaFlows framework. Then, the notions constituting the foundations of the framework are defined in Sect. 5.2. In Sect. 6, formal compliance notions are introduced. Sect. 7 considers the application of the framework and introduces a prototype developed in the SeaFlows project.

5.1 Fundamentals

Semantic constraints typically impose conditions on how processes should be carried out. Consider, for example, the constraints from Tab. 1. Constraints c_1 to

c_5, for example, impose conditions on the proper execution of patient treatment processes. Basically, they specify what must or must not happen during the execution of a treatment process.
From a technical point of view a process execution can be regarded as a sequence of process-related events (e.g., the execution of a particular activity). Hence, constraints on the proper process execution, in turn, can be regarded as constraints on the occurrences and the relations of such events. The SeaFlows framework makes use of this principle. In particular, the framework uses *event traces* to provide a *behavioural level view* on processes. This enables the specification of semantic constraints as constraints on the proper structure of event traces representing process executions.

The trace notion has already been applied to provide a logical model for dealing with process changes [17] and process evolution [13,58]. In the SeaFlows framework, the use of event traces allows for abstracting from concrete process meta-models. Hence, events and event traces can be regarded as interface between constraints and processes (cf. Fig. 6 (A)). This allows constraints, their formal semantics as well as formal compliance criteria to be developed extensively independent from concrete process meta-models. Furthermore, event traces are a suitable underlying logical model for an integrated support throughout the process lifecycle since they are applicable in all lifecycle phases. Fig 7 illustrates the behavioural level view and other concepts of the framework which build upon this view.

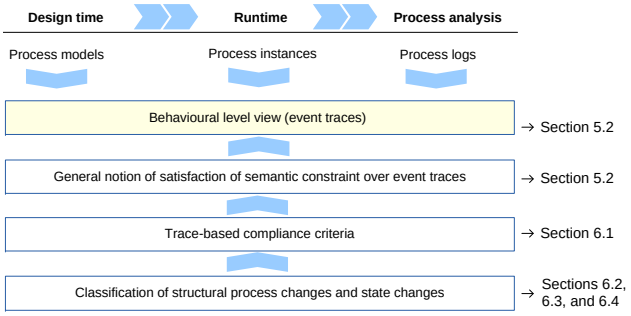


Fig. 7 Behavioural level view as basis for integrated support

Based on event traces, the framework introduces a general notion of satisfaction of semantic constraints which provides the basis for the trace-based compliance criteria for assessing the compliance of processes with semantic constraints. These criteria are applicable in all phases of the process lifecycle. Based on them, we provide formal criteria for classifying process changes and state changes (for compliance monitoring) with regard to their effects on the compliance with semantic constraints. Altogether, these criteria constitute an adequate basis for providing integrated lifecycle support.

5.2 Behavioural Level View for Constraint Integration

A basic concept common to all PrMS are *activities*. For a particular application domain (such as treatment processes in the healthcare domain) it is often possible to identify a set of activities relevant to processes of this domain. Activities typically operate on process context data (cf. Tab. 1). Activities of patient treatment (e.g., *inform patient*), for example, often require the patient's ID as input context. Similarly, a *tolerance test* may have several context data as outcome (e.g., the test results). The integration of such context data enables context-aware constraints. In this paper, we assume that context data relevant to constraints is available to the PrMS. Data integration issues will be part of our future research. Def. 1 provides a general notion of activities which also considers the activity context.

Definition 1 (Activity and activity execution) An *activity* a_t is defined as a 3-tuple

$(t, inputContext, outputContext)$ with:

- t denotes the activity type
- $inputContext = \{i_1, i_2, \ldots, i_n\}$ is the set of input context parameters of a_t
- $outputContext = \{o_1, o_2, \ldots, o_m\}$ is the set of output context parameters of a_t

Let a_t be an activity. Further, let $D^i_1, D^i_2, ..., D^i_n$ be the domains of the parameters in $inputContext$ of a_t and let $D^o_1, D^o_2, ..., D^o_m$ be the domains of the parameters in $outputContext$ of a_t. Then, an *activity execution* a is a 3-tuple

$(t, \{i_1(x_1), \ldots, i_n(x_n)\}, \{o_1(y_1), \ldots, o_m(y_m)\})$ with:

- $x_k \in D^i_k \cup \{undefined\},\ k = 1, \ldots, n$ and
- $y_l \in D^o_l \cup \{undefined\},\ l = 1, \ldots, m$

For a given domain, we denote as $\mathcal{A}$ the set of activities and as $\mathcal{A}^*$ the set of activity executions. □
An example of an activity execution is given below:

a = (Anamnesis, {PatId(mueller007)}, {HighRisk(false)})

a represents the execution of an anamnesis for a particular patient (with *PatId mueller007*). In addition, the patient's anamnesis has led to the result that he does not have an increased risk.

In a PrMS, activities themselves are subject to a lifecycle as well. In particular, they are undergoing different states triggered by events during process execution. Def. 2 formalises the notion of events employed by the SeaFlows framework.

Definition 2 (Event) Let $\mathcal{T}$ be the set of event types of interest. Let $\mathcal{A}^*$ be the set of activity executions for a given domain. An event e is a tuple $e = (t, a)$ with

- $t \in \mathcal{T}$ denotes the event type
- $a \in \$\mathcal{A}^*$ denotes the activity execution and its context data associated with e

We denote as $\mathcal{E}^*$ the set of all possible events for a given domain. □

Clearly, $\mathcal{T}$ determines the granularity of the behavioural view on processes. Event traces typically consist of `start` and `complete` events [3,58]. Based on traces using these event types, execution intervals of the corresponding activities can be obtained. The intervals, in turn, can be used to define temporal dependencies (e.g., temporal distance constraints and interval relations [42]). Generally, traces with `start` and `complete` events provide a good basis for defining the formal semantics of constraint specification formalisms since they can serve as expressive and general process representation. In practice, however, the event types `start` and `complete` can be too low-level for the *modelling* of constraints. In this case, constraint modelling can be facilitated by introducing abstractions from the underlying event trace model. In particular, respective constraint specification approaches should provide abstract concepts (e.g., by defining abstract predicates) such that constraint modellers do not have to directly deal with `start` and `complete` events.

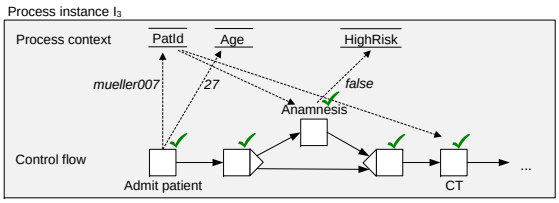
In this paper, we focus on providing general trace-based compliance criteria and not on introducing an expressive constraint specification formalism. For the purpose of presenting our ideas, the particular constraint types `start` and `complete` are not relevant. Hence, we opted for a more compact abstraction from `start` and `complete` events to illustrate our ideas. In particular, we will use the event type of performing an activity (i.e., $\mathcal{T} = \{\texttt{execute}\}$) in the following. The ordering relation of the `execute` events can be derived from a trace with `start` and `complete` events by, for example, taking the `start`/`start` relation of the activities as basis, respectively. However, note that the ideas presented in this paper are also applicable to event traces with `start` and `complete` event types.

Since we focus on the event type `execute` in this paper, we will use an abbreviation for denoting events in the following. Instead of `(execute, (Anamnesis,...,...))`, for example, we omit the event type and write `(Anamnesis,...,...)`. Due to the use of *activity executions* in events, the process context is also captured in an event. This extension is important as it enables the specification and evaluation of context-aware semantic constraints.

Clearly, an *event trace* is suitable for representing the execution of a process instance. A set of event traces, in turn, can be used as an abstraction from processes (cf. Fig. 7). Formally, an event trace is an ordered sequence of trace entries which, in turn, are assigned to events (cf. Def. 3).

Definition 3 (Event trace) An event trace $\sigma = <e_1,...,e_n>$ is an ordered sequence of trace entries e_i in which each entry $e_i \in \mathcal{E}^*$ corresponds to an event. Further, we denote by $\Sigma_{\mathcal{E}^*}$ the set of all event traces over $\mathcal{E}^*$. □

Fig. 8 provides an example of a process execution and the corresponding event trace.



Event trace = < e_1, e_2, e_3, e_4, e_5, ...> with
e_1= (**AdmitPatient**, {}, {PatId(mueller007),Age(27)})
e_2= (**Anamnesis**, {PatId(mueller007)}, {HighRisk(false)})
e_3= (**CT,** {PatId(mueller007)},{})
...

Fig. 8 A process instance and the corresponding event trace

Note that the information captured in the events and in the event traces is similar to common log formats like MXML as used by the process analysis framework ProM [3]. In particular, MXML has a designated data field for each event to store context data.

The notions of events and event traces provide the basis for constraint specification in the SeaFlows framework. In particular, a constraint (e.g., from Tab. 1) can be specified by defining rules on the proper occurrences of respective events and associated context.

In practice, it is often desirable to support the abstraction of concrete activities. Consider, for example, constraint c_5. In the healthcare domain, there can be a broad range of activities which are regarded as invasive procedure (e.g., a punction, a surgery, or an endoscopy). The specification of constraints corresponding to c_5 based on events for each of these activities can become a quite laborious task. The same applies to the maintenance of the respective constraints. Hence, to increase the ease of constraint specification it can be desirable to introduce the abstract concept *invasive procedure* (e.g., as an abstract activity) which subsumes the respective activities. Thereby, a constraint corresponding to c_5 can be specified by using events associated with the abstract activity *invasive procedure*. Such additional concepts and their relations can be modelled

using techniques from ontological engineering (e.g., using descriptions logic [10] or OWL [25]). This additional knowledge can be stored in a *knowledge base*. The SeaFlows framework supports such abstractions by introducing an optional knowledge base. The knowledge base can be integrated to further ease the procedure of constraint specification and maintenance. The detailed conception of the knowledge base is beyond the scope of this paper. In the remainder of this paper, we assume the existence of a single knowledge base.

Fig. 9 shows the relations of the concepts introduced so far.

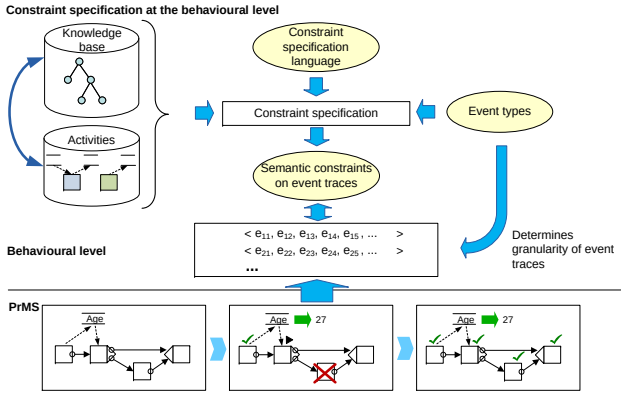


Fig. 9 Using the behavioural level view on processes for constraint specification

As depicted in Fig. 9, semantic constraints can make use of the activities and the concepts from the knowledge base (if available). The events serve as artefacts of semantic constraints. Based on the notion of event traces, we provide an abstract notion of satisfaction of semantic constraints. In particular, we abstract from concrete constraint syntax and focus on the semantics of a constraint over an event trace. Basically, the semantics of a semantic constraint can be regarded as a function assigning a boolean value (i.e., `true` or `false`) to an event trace with `true` indicating that the execution trace satisfies the constraint and `false` indicating the opposite.

Definition 4 (Satisfaction of semantic constraints) Let $\sigma \in \Sigma_{\mathcal{E}^*}$ be an event trace. Let $\mathcal{C}$ be the set of all semantic constraints and let $c \in \mathcal{C}$ be a semantic constraint. Then, the semantics of c is defined as a function *sat* with:

$$sat : \mathcal{C} \times \Sigma_{\mathcal{E}^*} \mapsto \{true, false\} \text{ with}$$

$$sat(c, \sigma) = \begin{cases} true \text{ if } \sigma \text{ satisfies } c \\ false \text{ otherwise} \end{cases}$$

- For $sat(c, \sigma) = true$, σ is a model for c (formally $\sigma \models c$).
- For $sat(c, \sigma) = false$, σ is not a model for c (formally $\sigma \not\models c$).

□

Note that it is not our objective to propose a particular constraint specification language. The behavioural level view provides a suitable logical model for a variety of specification languages (cf. Fig. 9). Since Def. 4 focuses on the constraint semantics it also does not imply a particular constraint specification language. Thus, the formal framework can be applied to any specification language which provides formal semantics corresponding to Def. 4.

5.3 Example

In the following, constraints c_2 and c_5 from Tab. 1 are specified using first order predicate logic (PL1) [33]. Constraint c_2 states that prior to each examination of a patient p who is older than 75, a tolerance test for patient p needs to take place. Constraint c_5 expresses that prior to each event associated with an invasive procedure for patient p, another event associated with informing patient p has to take place. Predicate *IsOfType* infers whether an event is associated with the given activity. Predicate $Pred(e_1, e_2)$ infers whether event e_1 is the predecessor of event e_2 in the event trace.

c_2:
$\forall e_1, p, a \;\; (IsOfType(e_1, examination)$
$\wedge\; PatientContext(e_1, p) \wedge AgeContext(e_1, a)$
$\wedge\; a \geq 75 \;\rightarrow$
$\quad \exists e_2 \;\; IsOfType(e_2, toleranceTest)$
$\quad \wedge PatientContext(e_2, p) \wedge Pred(e_2, e_1))$

c_5:
$\forall e_1, p \;\; (IsOfType(e_1, invasiveProcedure)$
$\wedge\; PatientContext(e_1, p) \;\rightarrow$
$\quad \exists e_2 \;\; IsOfType(e_2, informPatient)$
$\quad \wedge PatientContext(e_2, p) \wedge Pred(e_2, e_1))$

Additionally, a knowledge base containing relations between activities is given. It relates the activities *surgery*, *endoscopy*, and *punction* to the abstract activity *invasive procedure*:
$\forall e \; (IsOfType(e, surgery) \;\rightarrow$
$\quad IsOfType(e, invasiveProcedure))$
$\forall e \; (IsOfType(e, punction) \;\rightarrow$
$\quad IsOfType(e, invasiveProcedure))$
$\forall e \; (IsOfType(e, endoscopy) \;\rightarrow$
$\quad IsOfType(e, invasiveProcedure))$

A given event trace and the knowledge base yield an *interpretation* of the formulas. Due to the clauses in the

knowledge base, events associated with the activities *surgery*, *punction*, or *endoscopy* will match with the specifications concerning the *invasive procedure* when evaluating constraint c_5.

The use of PL1 for constraint specification only serves as example. The rationale behind it is that PL1 is intelligible and commonly known. A discussion on which constraint specification language is suitable for which applications is beyond the scope of this paper and part of future research.

6 Formal Compliance Criteria for Lifecycle Support

With the introduction of the behavioural level view on processes, we have provided a formal basis for developing compliance criteria. The latter are addressed in the following. Moreover, we take a step further by introducing classification criteria for evaluating structural process changes and state changes with regard to their effects on the compliance with semantic constraints. Altogether, these criteria provide the basis for realizing integrated compliance support throughout the process lifecycle.

6.1 On Assessing the Compliance with Semantic Constraints

Taking on the definition of semantic constraints (Def. 4) we can derive satisfaction notions for a *set of event traces* as follows:

Definition 5 (Satisfaction of constraints over event traces) Let c be a semantic constraint and let $\Sigma=\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ be a set of traces. Then, we define the following satisfaction criteria:

- c is *violated* over Σ:
  $\Leftrightarrow \forall\ \sigma_i$, i = 1, ..., n, $\sigma_i \not\models c$ holds
  Formally: $violated(c, \Sigma)$
- c is *satisfied* over Σ:
  $\Leftrightarrow \forall\ \sigma_i$, i = 1, ..., n, $\sigma_i \models c$, i = 1, ..., n, holds
  Formally: $satisfied(c, \Sigma)$
- c is *violable* over Σ:
  $\Leftrightarrow \neg violated(c, \Sigma)$ and $\neg satisfied(c, \Sigma)$
  Formally: $violable(c, \Sigma)$

□

The satisfaction notions from Def. 5 provide a basis for assessing the compliance of process models, process instances, and process executions logs with imposed semantic constraints. How this can be accomplished is illustrated in Fig. 10. For a process model, the traces

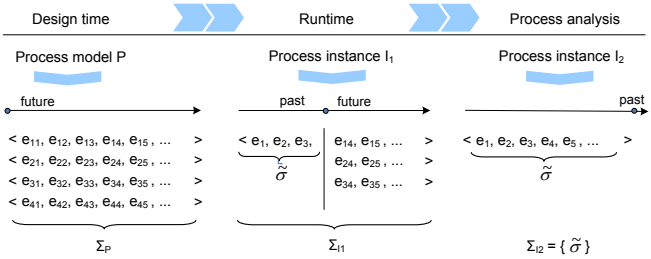


Fig. 10 Process lifecycle and event traces

which can be derived from it represent the possible process behaviour. Therefore, these execution traces are relevant to compliance checks. For a process instance, in turn, future behaviour may also depend on its past behaviour (i.e., $\tilde{\sigma}$). This has to be taken into account when assessing the compliance of process instances with semantic constraints. In order to accomplish an a posteriori analysis of a completed process execution, we basically will have to check its event history $\tilde{\sigma}$ for compliance. These considerations are formalised in Def. 6.

Definition 6 (Compliance of processes with a semantic constraint)
(A) Let P be a process model and let $\Sigma_P=\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ be the set of traces which can be derived from P. Let c be a semantic constraint imposed on P. Then:

- $violated(c, P) \Leftrightarrow violated(c, \Sigma_P)$
- $satisfied(c, P) \Leftrightarrow satisfied(c, \Sigma_P)$
- $violable(c, P) \Leftrightarrow violable(c, \Sigma_P)$

(B) Let $I = (P, \tilde{\sigma})$ be a process instance with P being the process model and $\tilde{\sigma}$ being the current event history of I. Let $\Sigma_{I,\tilde{\sigma}}=\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be the set of traces which can be derived from I with $\Sigma_{I,\tilde{\sigma}}=\{\sigma \in \Sigma_P | \exists\ \bar{\sigma} \in \Sigma_{\mathcal{E}^*}\ with\ \sigma = \tilde{\sigma}\bar{\sigma}\}$[1]. Further, let c be a semantic constraint imposed on I. Then:

- $violated(c, I) \Leftrightarrow violated(c, \Sigma_{I,\tilde{\sigma}})$
- $satisfied(c, I) \Leftrightarrow satisfied(c, \Sigma_{I,\tilde{\sigma}})$
- $violable(c, I) \Leftrightarrow violable(c, \Sigma_{I,\tilde{\sigma}})$

□

These trace-based criteria provide the formal means for assessing the compliance of a process with a semantic constraint at any process lifecycle phase. Moreover, Def. 6 also provides the foundations for classifying and evaluating *process changes*. This becomes necessary for compliance monitoring as well as for validating process changes (cf. Req. 4.2 and 4.3). In Sect. 6.2, we first derive formal criteria for evaluating state changes. Then, we address structural process changes in Sect. 6.3. In Sect. 6.4, we give an idea of how to apply these criteria.

[1] $\tilde{\sigma}\bar{\sigma}$ denotes the concatenation of $\tilde{\sigma}$ and $\bar{\sigma}$

The introduced criteria provide the formal foundations for monitoring process instance executions and for validating process changes at process model and at process instance level.

6.2 On Dealing with State Changes

As motivated, a constraint-aware PrMS has to provide adequate mechanisms for compliance monitoring. These particularly will become necessary if not all constraints are enforced at process model level (cf. Sect. 2.2). In this case, the PrMS has to monitor the execution progress of process instances (i.e., the execution history of process instances including their context) and trigger validation mechanisms if necessary. Since changes to the execution progress of a process instance can be considered as changes to its *state*, we refer to them as *state changes*. In this paper, we only consider state changes evolving within a process instance. This means that state changes caused by concurrently running instances are not considered.

Def. 7 formalises the notion of state changes. Informally, a state change corresponds to changes of the event history of the process instance, whereas the underlying process model P remains unchanged.

Definition 7 (State change) Let $I = (P, \tilde{\sigma})$ be a process instance with $\Sigma_{I,\tilde{\sigma}}=\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ being the set of traces which can be generated from I with respect to $\tilde{\sigma}$. Then, a state change Δ_e on I with respect to $\tilde{\sigma}$ is defined as:
$\Delta_e \in \mathcal{E}^*$ with $\exists\ \sigma_i \in \Sigma_{I,\tilde{\sigma}}$ with $\sigma_i = \tilde{\sigma}\Delta_e\breve{\sigma}$[2], $\breve{\sigma} \in \Sigma_{\mathcal{E}^*}$

Further, we denote as $I[\Delta_e > I'$ the application of the state change Δ_e to I yielding process instance I' with $I' = (P, \hat{\sigma})$ and $\hat{\sigma} = \tilde{\sigma}\Delta_e$[3]. □

A state change may have different effects on the satisfaction status of a constraint. Tab. 2 shows the possible effects of state changes with respect to the initial state of the constraint. In (A), c is initially satisfied over I. Then, the state changes do not have any effects on the satisfaction of c. The reason for this is that a state change Δ_e basically corresponds to a selection over the set of the potential traces of I (i.e., $\Sigma_{I,\tilde{\sigma}}$). The same will apply, if the constraint is initially violated over I. These considerations have impact on the practical realization of a constraint-aware PrMS. In particular, these considerations can be exploited and applied to constraint monitoring. If a semantic constraint is satisfied over a process instance, this instance will not require any monitoring with regard to this constraint. By contrast, if a semantic constraint is violated over a process model, it will not make sense to create instances from this model as the constraint will always be violated (unless structural changes happen). In case a semantic constraint is violable over a process, notions from Tab. 2 can be used to classify and monitor the constraint. If a state change led to the violation of a constraint (e.g., $violable(c, I)$ results in $violated(c, I')$), the administrator of the process instance should be notified and compensatory actions could be proposed.

Example Fig. 11 illustrates how the classification above can be applied to monitor state changes. In process instance I_4, it is not determined yet whether or not the CT examination will be carried out with non-water-soluble contrast agents. Hence, constraint c_3 from Tab. 1 is *violable* over I_4. By contrast, the execution of process instance I'_4 has proceeded further (due to the completion of the CT examination). According to Δ_e, which represents the execution of the CT examination, non-water-soluble contrast agents have been used for the examination. Hence, the state change denoted by Δ_e has impact on the satisfaction of c_3. In particular, c_3 is *violated* over I'_4 (assuming that the CT and the endosonography are scheduled for the same week). Therefore, Δ_e is a *violating* state change.

6.3 On Dealing with Structural Process Changes

As pointed out, business processes may be subject to change (e.g., due to process optimization). In particular, this necessitates modifications of the process structure (i.e., structural change). Process changes may take place at process model or process instance level. To avoid errors, process changes need to be carried out in a controlled manner [57]. This does not only apply to structural aspects but also to compliance with semantic constraints. Hence, mechanisms are required to evaluate process changes with regard to compliance with imposed constraints. For this purpose, we provide a classification of process changes with regard to their effects on the satisfaction of semantic constraints.

Def. 8 introduces a formal notion of structural process changes with regard to their *effects* on the process behaviour. A process change (i.e., Δ_I) consists of a *change operation*. Typical elementary change operations are `insert`, `delete`, and `move` [67]. However, Def. 8 abstracts from concrete change operations and focuses on their effects instead. Informally, Δ_I modifies the underlying process model P.

Definition 8 (Structural process change) Let Δ_* be a set of process change operations with regard to

[2] $\tilde{\sigma}\Delta_e\breve{\sigma}$ denotes the concatenation of $\tilde{\sigma}$, Δ_e, and $\breve{\sigma}$
[3] $\tilde{\sigma}\Delta_e$ denotes the concatenation of $\tilde{\sigma}$ and Δ_e

Table 2 Classification of state changes

	Initial constraint state	Resulting situations ($I[\Delta_e > I' = (P, \hat{\sigma})$)
(A)	$satisfied(c, I)$	$satisfied(c, I')$ (Δ_e has no effect on the satisfaction of c)
(B)	$violated(c, I)$	$violated(c, I')$ (Δ_e has no effect on the violation of c)
(C)	$violable(c, I)$	$satisfied(c, I')$ (Δ_e is *healing* with regard to c) $violated(c, I')$ (Δ_e is *violating* with regard to c) $violable(c, I')$ (Δ_e is *neutral* with regard to c)

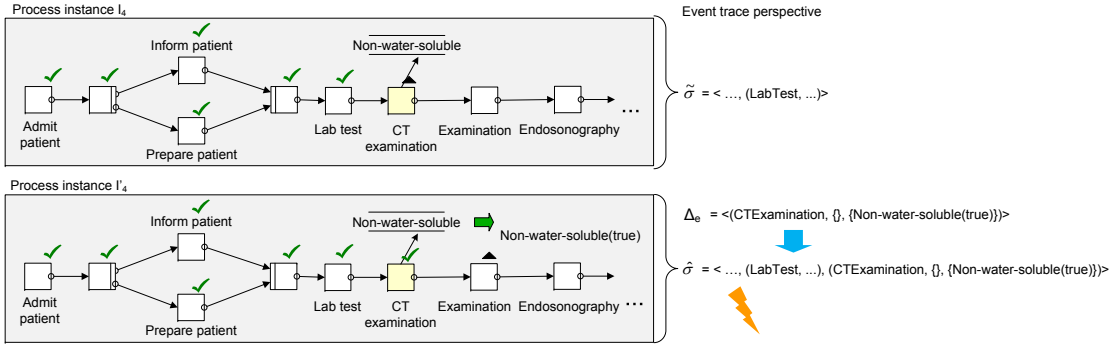


Fig. 11 Applying the formal criteria for monitoring state changes

a set of activities and corresponding context data of a particular domain. Further, let P be a process model. Then, a process change $\Delta_P = <\Delta_{P,1}, \ldots, \Delta_{P,l}>$ with $\Delta_{P,i} \in \Delta_*$, $i = 1, \ldots, l$ is defined as a sequence of change operations.

Further, we denote as $P[\Delta_P > P'$ the application of Δ_P to P yielding process model P' □

Note that a *process instance change* Δ_I can be regarded as a special case of process model changes with $I = (P, \tilde{\sigma})$ and $I[\Delta_I > I' = (P', \tilde{\sigma})$. Hence, we abstain from providing a separate definition for process instance changes.

Tab. 3 shows the possible effects of structural process changes with respect to the initial state of the constraint. In contrast to state changes, which can be regarded as a selection over the premodelled process behaviour, structural process changes are capable of really modifying the process behaviour. How these change effects can be applied to evaluate process changes is shown in the following.

Example In Fig. 12, process instance I_5 is adapted by inserting a *punction* after the *examination*. This results in process instance I'$_5$. Based on the traces which can be derived from I'$_5$ the insertion of the *punction* can be classified as a *violating* change. Hence, the process engineer may undo the change to restore compliance with c_5. Being aware of the violation, the process engineer may also apply further adaptations. Fig. 12 shows two alternative subsequent process changes to I'$_5$. The application of Δ_A would yield a process instance which is violable with regard to c_5 (A). Hence, Δ_A is a *partially healing change*. The application of Δ_B, in turn, would yield a process instance which is satisfied with regard to c_5 (B). Hence, Δ_B is a *healing change*.

The criteria for evaluating structural process changes are general and therefore are not restricted to a particular change framework. However, the criteria rely on a structural process change framework which ensures the *correctness* of process changes with regard to structural properties. For example, for ensuring that process changes do not lead to inconsistencies such as introducing deadlocks or changing the past [59].

6.4 On Dealing with Violations

In many cases in practice, it is not sufficient to solely find out whether or not a constraint is violated over a process. In order to assist process engineers and process performers to deal with violations, it becomes necessary to also find out whether a violation is *healable*. Consider for example Fig. 13. Process instances I_6 and I'$_6$ both violate constraint c_2. Due to the execution state of I_6 the violation is still healable (e.g., by inserting a tolerance test). In I'$_6$, however, the violation of c_2 cannot be avoided anymore since the examination has already been carried out[4]. Def. 9 provides a formalisation of the notion of healable violations.

[4] We assume that reparational actions are also incorporated into the constraints as it is possible with FCL [28,26]

Table 3 Classification of structural process changes

	Initial constraint state	Resulting situations ($I[\Delta_I > I' = (P', \tilde{\sigma})$)
(A)	$satisfied(c, I)$	$satisfied(c, I')$ (Δ_I is *neutral* with regard to c) $violated(c, I')$ (Δ_I is *violating* with regard to c) $violable(c, I')$ (Δ_I is *partially violating* with regard to c)
(B)	violated(c,I)	satisfied(c,I') (Δ_I is *healing* with regard to c) $violated(c, I')$ (Δ_I is *neutral* with regard to c) $violable(c, I')$ (Δ_I is *partially healing* with regard to c)
(C)	$violable(c, I)$	$satisfied(c, I')$ (Δ_I is *healing* with regard to c) $violated(c, I')$ (Δ_I is *violating* with regard to c) $violable(c, I')$ (Δ_I is *neutral* with regard to c)

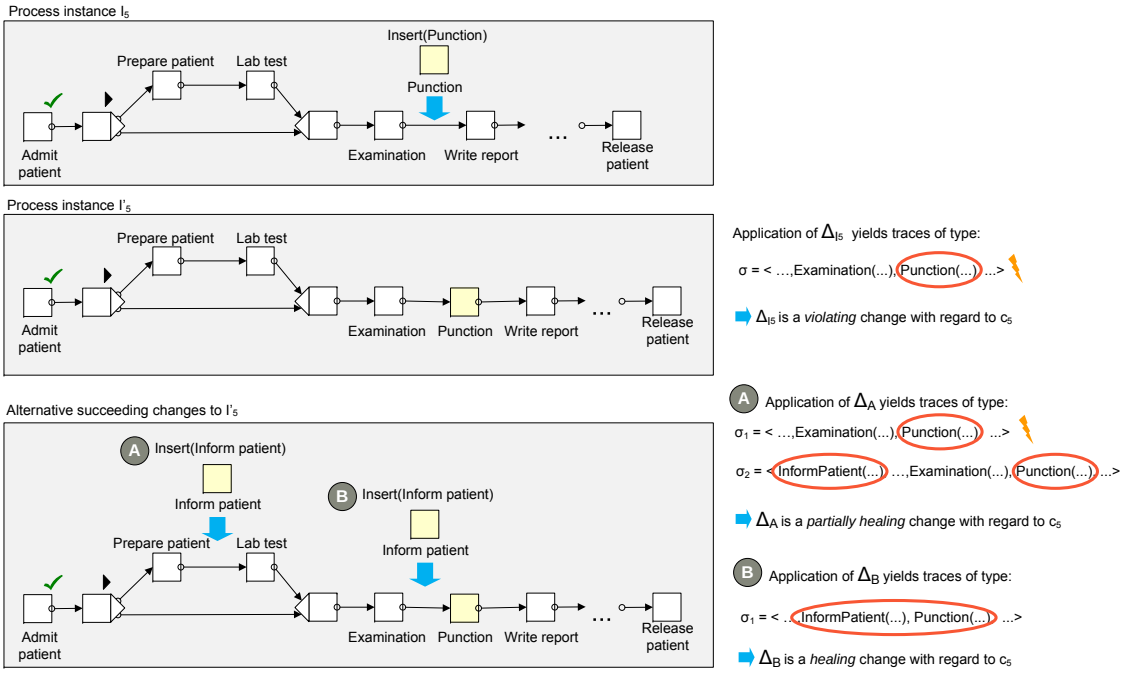


Fig. 12 Process changes and their effects on compliance

Definition 9 (Healable violations) Let c be a semantic constraint. Further, let $I = (P, \tilde{\sigma})$ be a process instance and let c be violated over I (i.e., $violated(c, I)$). Then:
$healable(c, I) \Leftrightarrow$
$\exists\ \Delta_I$ with $I[\Delta_I > I'$ and $satisfied(c, I')$ holds □

According to Def. 9 a violation will be healable if and only if there exists a set of process change operations which can be properly applied to the process instance such that the resulting instance satisfies the constraint. Note that a constraint which is violable over a process instance I (i.e., $violable(c, I)$ is true) is *always* healable over I. This is because it is possible to create a set of change operations which removes exactly those traces violating the constraint (i.e., traces σ with $\sigma \not\models c$). This can lead to restructuring the process or inserting additional branches. Def. 9 can be adapted for process models as well. Note that a violation introduced by a structural process change is always trivially healable according to Def. 9. This can be achieved by undoing the structural change.

The notion of healable violations can be exploited for monitoring process instances (i.e., monitoring of state changes). For this purpose, when an user interaction is requested (such as starting an activity), it is first necessary to relate the requested interaction to corresponding state changes. This allows for compliance validation with regard to the state changes. In case a violation occurs, it has to be assessed whether the violation is healable. Clearly, non-healable violations are more severe than healable ones. Hence, the strategy for dealing with violations should take this into account. However, when monitoring process executions, it also has to be considered whether interactions leading to

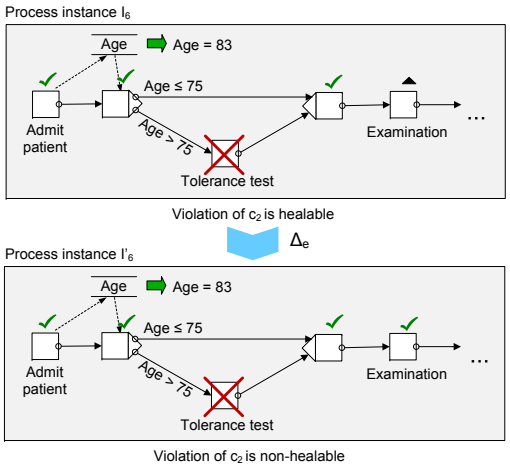


Fig. 13 Healable and non-healable constraint violations

non-healable state changes are rejectable or not. For example, the results of a lab test are not rejectable whereas the administration of a particular drug can be rejected. This particularly depends on the semantics of the events but also on other aspects such as time of events and real execution times. However, employing the notion of healable violations and rejectable interactions would allow for the definition of sophisticated strategies for dealing with violations.

7 On Applying the SeaFlows Framework

The main objective of the SeaFlows framework is to provide formal backgrounds which can be used to build concrete approaches for supporting semantic constraint upon. The criteria introduced constitute the formal foundations for assessing the compliance of processes and process changes in different phases of the process lifecycle. Due to being based on the notion of event traces they are general and can be applied to a broad range of process meta-models.

However, the trace-based criteria are not supposed to be directly implemented as operational level checks. This is often not feasible in practice due to efficiency reasons. The exploration of event traces is often too costly to be carried out in practice. How compliance checking approaches are realized in practice highly depends on the constraint specification approach as well as on the process meta-model. Nevertheless, there are general optimizations which are useful for practical realization of compliance checks based on formal framework. These optimizations are presented in Sect. 7.1. Then, we present the SeaFlows prototype which makes use of some of these considerations in order to enable efficient validation and helpful validation reports.

7.1 Optimizations for Practical Application

Operational level checks The exploration of event traces is not feasible for practical application, especially when dealing with complex processes. Moreover, we often have to deal with infinite sets of event traces (e.g., process models with loops or context parameters with an infinite domain). To handle this issue, *abstraction techniques* can be applied. For example, the domain of the context parameter age obviously can be large or infinite. However, if it is only relevant to constraints whether or not the patient is older than 75, we will be able to abstract from the concrete values and to focus on the only two relevant cases (i.e., older than 75 and the opposite case). Such abstraction techniques are widely used for model checking [62]. Clearly, the adoption of abstraction techniques is a viable strategy for dealing with state explosions.

Context-sensitive checks For monitoring process executions and for validating process changes, it is not necessary to recheck all constraints. Here, the semantics of the constraint as well as the semantics of the change can be exploited in order to decide which constraints may be affected by the change. Based on the change semantics, we can, for example, identify *neutral* changes (cf. Tab. 3). In previous work [44], we already proposed a first approach to identify potentially affected constraints. This approach will be further extended to deal with more complex constraints. This will help to enable more efficient runtime validation.

Constraint-specific refinements Due to being general, the formal framework treats semantic constraints as a black box. Hence, a further optimization could be achieved by exploiting knowledge about the proper structure of semantic constraints.

7.2 The SeaFlows Prototype

We have implemented some of the presented ideas in the SeaFlows prototype. At present, the prototype enables the specification of complex execution dependencies between activities. Constraints can be specified over activities available in the domain's *activity repository* and stored in constraint files. Multiple constraint files can be assigned to a process model. In addition, the prototype supports the validation of ADEPT process models against the assigned constraints.

The validation is not carried out by trace exploration of process models (i.e., behavioural verification). Instead, we perform checks at the structural level (i.e., by checking the process structure which determines

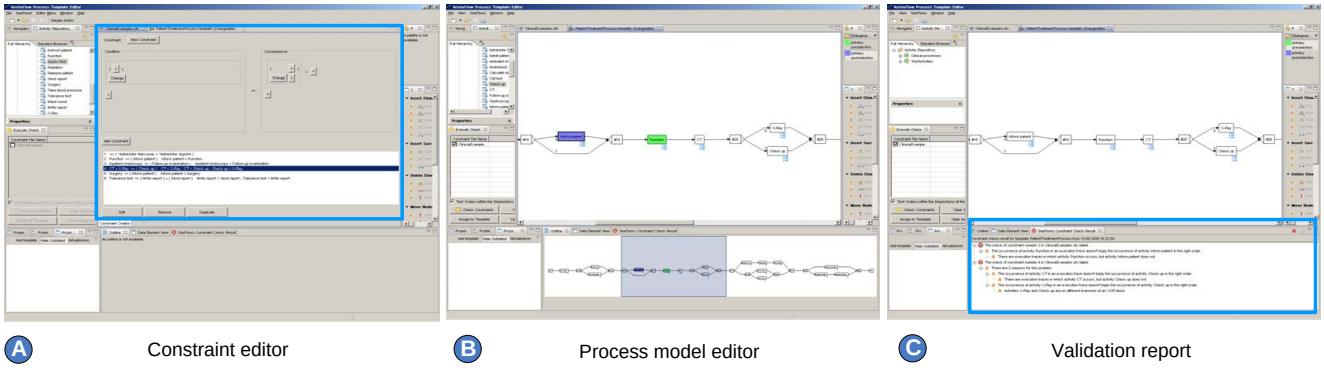


Fig. 14 User interfaces of the SeaFlows prototype

the behaviour). This done by automatically deriving *structural* correctness criteria from given execution constraints and by checking them for ADEPT process graphs, respectively. Let us explain the basic idea using a small example: if, for example, an existence dependencies exists between two activities A and B (i.e., B always needs to be executed after A), then this dependency will only be satisfied if either A never occurs in the process (trivial satisfaction) or there exists B in the process model such that the execution of B is not optional to A. For validating such structural criteria, mechanisms similar to data flow analysis (e.g., for ensuring the proper supply of data parameters [57]) can be applied. This structural checking of ADEPT processes allows for more efficient validation. In addition, it also allows for providing sophisticated feedback in case of constraint violations.

The SeaFlows prototype is integrated into the AristaFlow BPM Suite [5] by exploiting the Eclipse plug-in framework. Fig. 14 shows the prototype's user interfaces: The constraint editor (A), the AristaFlow process model editor into which the prototype is integrated (B), and the validation report (C). Fig. 15 shows the process validation in more detail. We have modelled two semantic constraints (A). The first constraint expresses that the activity *inform patient* has to take place prior to the activity *punction*. The second constraint expresses that a *check-up* has to be carried out between the execution of the activities *CT* and *endosonography*. As shown in Fig. 15, the prototype provides a detailed validation report (C). The process from Fig. 15 (B) is not compliant with constraint 1 since *inform patient* is optional to the execution of *punction*. Further, the process violates constraint 2 since the *check-up* is optional to the *CT* and exclusive to the *endosonography*. The information provided by the report can be exploited by the process engineer to modify the process model accordingly (e.g., by moving the *check-up* such that it is not optional to *CT* and not exclusive to the *endosonography*).

At present, our prototype does not support the full range of features envisaged for the framework. Advanced features such as runtime monitoring for context-aware constraints are still to be implemented. Nevertheless, with the implementation of this prototype, we have taken the initial steps for the technical realization of the SeaFlows framework. The prototype will be updated in order to integrate newer concepts and results from the SeaFlows project. In future work, we will also consider the application of propagation algorithms as proposed in [27, 70, 69]. These propagation mechanisms can be useful particularly when dealing with context-aware constraints.

8 Summary and Outlook

Business process compliance has become a big issue for today's organisations and a challenge for process-oriented information systems in general. Due to the heterogeneous IT landscapes in organisations and cross-organisational processes, business process compliance management often has to deal with complex scenarios [36]. Enabling PrMS to support the compliance of processes with imposed semantic constraints can be regarded as one step towards the installation of business process compliance management in practice.

Doubtlessly, compliance requirements have also led to new requirements on process management system technology. In this paper, we first elaborated on fundamental requirements for supporting semantic constraints in PrMS. In addition, we provided an overview on existing approaches and discussed to what extent they are able to meet the requirements. We showed that there is a demand for an approach allowing for integrated compliance support with regard to the process lifecycle. In this paper, we addressed the particular challenge of enabling life time compliance. In particular,

[5] www.aristaflow.com
www.uni-ulm.de/en/in/iui-dbis/research/projects/adept2.html

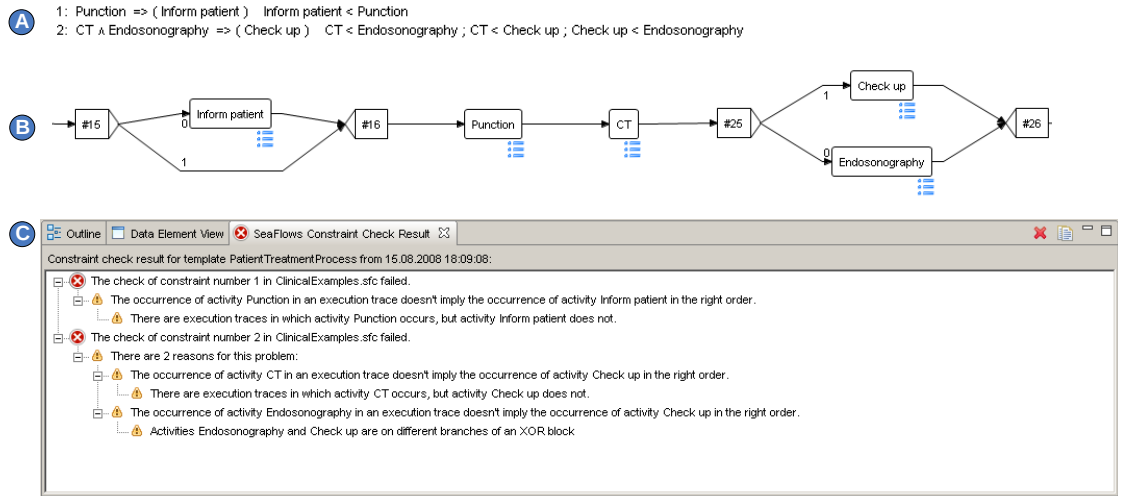


Fig. 15 Process validation using the SeaFlows prototype

we introduced the SeaFlows framework which makes two fundamental contributions. Firstly, it introduces a behavioural level view on processes which serves a logical model for compliance criteria applicable throughout the process lifecycle. This enables the specification of constraints as well as compliance criteria abstracting from concrete process meta-models. Secondly, the framework provides formal trace-based compliance criteria not only for static compliance validation but also for dealing with process changes (in particular, state changes and structural changes). These criteria employ a graded notion of compliance and can serve as formal basis for the implementation of compliance checks. We also presented our prototype which is based on the concepts of the framework. Altogether, the framework proposed in this paper can serve as formal foundation for developing constraint-aware PrMS.

Clearly, there are also many questions left to future research. One important challenge is the integration of ontological concepts for ease of constraint specification and management. Here, interesting results from existing approaches on semantic process validation [27, 70] and on domain modelling from AI research will also be considered. In addition, the development of efficient operational level checks for assessing compliance is also an important issue. This is part of ongoing research. We will also investigate how to deal with context changes caused by external events. In the SeaFlows framework, we focused on a rather technical perspective. To be able to go all the way from high level constraint specification to implementation level constraints also requires further research. Here, we can benefit from ideas and concepts from the business rules research (e.g., the SBVR standard [54]). These will have to be further investigated in more detail to be adopted for supporting semantic constraints in PrMS. A further challenge is to tackle the multiple process scenario. One idea is to merge the (partial) histories of concurrently running process instances which are within the scope of the same constraint (e.g., multiple treatment processes for the same patient). Thus, these processes could be treated as one process execution at the formal level. This idea still needs further investigations. Moreover, strategies for validating and monitoring concurrent processes at the operational level still have to be developed.

In fact, the support of semantic constraints in PrMS is a cross-cutting concern. Hence, related research fields such as case handling, ontological engineering, multi-agent systems, and enterprise knowledge management can also provide further solutions and inspiration for future developments.

Acknowledgements We would like to thank our students Heiko Fröhlich, Philipp Merkel, Barbara Panzer, and Andreas Pröbstle for implementing the SeaFlows prototype presented in this paper.

References

1. Directive 2006/43/EC of the European Parliament and of the Council of 17 May 2006 (2006). URL http://www.eur-lex.europa.eu
2. van der Aalst, W.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: Proc. BPM '00, pp. 161–183 (2000)
3. van der Aalst, W., et al.: Prom 4.0: Comprehensive support for real process analysis. In: Proc. of Application and Theory of Petri Nets and Other Models of Concurrency, *LNCS*, vol. 4546, pp. 484–494. Springer (2007)
4. van der Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoret. Comp. Science **270**(1-2), 125–203 (2002)

5. van der Aalst, W., de Beer, H., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: Proc. OTM Conferences '05, *LNCS*, vol. 3761, pp. 130–147 (2005)
6. Agrawal, R., Johnson, C., Kiernan, J., Leymann, F.: Taming compliance with Sarbanes-Oxley internal controls using database technology. In: Proc. of the 22nd Int'l Conf. on Data Engineering (ICDE'06), p. 92. IEEE Computer Society (2006)
7. Alberti, M., et al.: Computational logic for run-time verification of web services choreographies: Exploiting the SOCS-SI tool. In: Proc. 3rd Int'l Workshop on Web Services and Formal Methods, *LNCS*, vol. 4184, pp. 58–72. Springer (2006)
8. Alberti, M., et al.: Expressing and verifying business contracts with abductive logic. In: Normative Multi-agent Systems, no. 07122 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (2007)
9. Attie, P., Singh, M., Sheth, A., Rusinkiewicz, M.: Specifying and enforcing intertask dependencies. In: Proc. VLDB '93, pp. 134–145. Morgan Kaufmann Publishers Inc. (1993)
10. Baader, F., et al.: The Description Logic Handbook - Theory, Implementation and Applications. Cambridge University Press (2003)
11. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (2002)
12. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the visualization of system-spanning business processes. In: Proc. 1st Int'l Workshop on Business Process Monitoring and Performance Management (BPMPM'05) (2005)
13. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. Data & Knowledge Engineering **24**(3), 211–238 (1998)
14. Dadam, P., et al.: Towards truly flexible and adaptive process-aware information systems. In: Proc. UNISCON '08, Klagenfurt, Austria (2008)
15. Davulcu, H., Kifer, M., Ramakrishnan, C.R., Ramakrishnan, I.V.: Logic based modeling and analysis of workflows. In: PODS '98, pp. 25–33 (1998)
16. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proc. of the 21st Int'l Conf. on Software Engineering, pp. 411 – 420 (1999)
17. Ellis, C., Keddara, K., Rozenberg, G.: Dynamic change within workflow systems. In: Proc. of the Int'l ACM Conf. COOCS '95, pp. 10–21 (1995)
18. Emprise: BONAPART Sarbanes-Oxley Analyzer (2008)
19. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based analysis of obligations in web service choreography. In: Proc. of the Advanced Int'l Conf. on Telecommunications and Int'l Conf. on Internet and Web Applications and Services, p. 149. IEEE Computer Society (2006)
20. Förster, A., Engels, G., Schattkowsky, T., Van der Straeten, R.: Verficiation of business process quality constraints based on visual process patterns. In: Proc. 1st Joint IEEE/IFIP Symposium on Theoretical Aspects of Sofware Engineering (2007)
21. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and verifying workflow-based regulations. In: Workshop on Regulations Modelling and their Validation and Verification (2006)
22. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Proc. ICSOC '07, *LNCS*, vol. 4749, pp. 169–180. Springer (2007)
23. Giblin, C., Müller, S., Pfitzmann, B.: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Tech. Rep. Research Report RZ-3662, IBM Research GmbH (2006)
24. Goldszmidt, G., Joseph, J., Sachdeva, N.: On demand business process life cycle, part 6: Apply customization policies and rules. Tech. rep., IBM (2005)
25. Gomez-Perez, A., Fernandez-Lopez, M., Corcho-Garcia, O.: Ontological Engineering. Springer (2004)
26. Governatori, G.: Representing business contracts in RuleML. Int. J. Cooperative Inf. Syst. **14**(2-3), 181–216 (2005)
27. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: Proc. of the 4th Int'l Workshop on Business Process Design (2008)
28. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Proc. EDOC'06, pp. 221–232. IEEE Computer Society (2006)
29. Greiner, U., Ramsch, J., Heller, B., Löffler, M., Müller, R., Rahm, E.: Adaptive guideline-based treatment workflows with adaptflow. In: Proc. of Symposium on Computerized Guidelines and Protocols, vol. 101, pp. 113–117. IOS Press (2004)
30. Heinlein, C.: Workflow and process synchronisation with interaction expressions and graphs. In: Proc. of the 17th Int'l Conf. on Data Engineering (ICDE '01). IEEE (2001)
31. Herbst, H., Knolmayer, G., Myrach, T., Schlesinger, M.: The specification of business rules: A comparison of selected methodologies. In: Methods and Associated Tools for the Information System Life Cycle, pp. 29–46. Elsevier (1994)
32. van den Heuvel, J., Weigand, H.: Cross-organizational workflow integration using contracts. In: Proc. Business Object Workshop '00 (2000)
33. Huth, M., Ryan, M.: Logic in Computer Science – Modelling and Reasoning about Systems. Cambridge University Press (2004)
34. IDS Scheer: Governance, risk and compliance management with ARIS (2008). In german
35. ILOG: ILOG JRules and IBM MQWF – White Paper (2005)
36. Kharbili, M.E., Stein, S., Markovic, I., Pulvermüller, E.: Towards a framework for semantic business process compliance management. In: Proc. of the 1st Int'l Workshop on Governance, Risk and Compliance (GRCIS'08), pp. 1–15 (2008)
37. Konyen, I., et al.: Process design for minimally-invasive surgeries. Interne Ulmer Informatik-Berichte DBIS-14, Ulm University (1996). In german
38. Lenz, R., Reichert, M.: It support for healthcare processes – premises, challenges, perspectives. Data & Knowledge Engineering **61**(1), 39–58 (2007)
39. Linehan, M., Ferguson, D.: Business rule standards – interoperability and portability. In: W3C Workshop on Rule Languages for Interoperability (2005)
40. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal **46**(2), 335–261 (2007)
41. Lu, R., Sadiq, S., Governatori, G.: Compliance aware process design. In: Proc. BPM Workshops '07, *LNCS*, vol. 4928, pp. 120–131. Springer (2008)
42. Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G.: Using a temporal constraint network for business process execution. In: Proc. of the 17th Australasian Database Conference, pp. 157–166. Australian Computer Society, Inc. (2006)
43. Ly, L.T., Rinderle, S., Dadam, P.: Semantic correctness in adaptive process management systems. In: Proc. BPM'06, *LNCS*, vol. 4102, pp. 193–208. Springer (2006)
44. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. Data & Knowledge Engineering **64**, 3–23 (2007)
45. Marchetti, A.M.: Sarbanes-Oxley Ongoing Compliance Guide: Key Processes and Summary Checklists. Wiley (2007)

46. Milosevic, Z., Josang, A., Dimitrakos, T., Patton, M.: Discretionary enforcement of electronic contracts. In: Proc. EDOC '02, pp. 3–14. IEEE Computer Society (2002)
47. Müller, D., Herbst, J., Hammori, M., Reichert, M.: IT support for release management processes in the automotive industry. In: Proc. BPM'06, *LNCS*, vol. 4102, pp. 368–377. Springer (2006)
48. Müller, R., Greiner, U., Rahm, E.: AgentWork: A workflow system supporting rule-based workflow adaption. Data & Knowledge Engineering **51**, 223–256 (2004)
49. zur Muehlen, M.: Organizational management in workflow applications – issues and perspectives. Inf. Tech. and Management **5**(3-4), 271–291 (2004)
50. zur Muehlen, M., Indulska, M., Kamp, G.: Business process and business rule modeling languages for compliance management: A representational analysis. In: ER (Tutorials, Posters, Panels & Industrial Contributions), pp. 127–132 (2007)
51. Namiri, K., Stojanovic, N.: A formal approach for internal controls compliance in business processes. In: 8th Workshop on Business Process Modeling, Development, and Support (2007)
52. Namiri, K., Stojanovic, N.: Pattern-based design and validation of business process compliance. In: OTM 2007, Part I, *LNCS*, vol. 4803, pp. 59–76. Springer (2007)
53. Newcastle Guideline Development and Research Unit: Management of dyspepsia in adults in primary care (2004)
54. OMG: Semantics of business vocabulary and business rules (sbvr), version 1.0 (2008). URL http://www.omg.org/spec/SBVR/1.0/PDF
55. Peleg, M., Soffer, P., Ghattas, J.: Mining process execution and outcomes – Position paper. In: BPM'07 Workshops, *LNCS*, vol. 4928, pp. 395–400. Springer (2008)
56. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: BPM'06 Workshops, *LNCS*, vol. 4103, pp. 169–180. Springer (2006)
57. Reichert, M., Dadam, P.: ADEPTflex – supporting dynamic changes of workflows without losing control. Journal of Intelligent Information Systems, Special Issue on Workflow Management Systems **10**(2), 93–129 (1998)
58. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – A survey. Data & Knowledge Engineering **50**(1), 9–34 (2004)
59. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases **16**(1), 91–116 (2004)
60. Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: Proc. BPM'07, *LNCS*, vol. 4714, pp. 149–164. Springer (2007)
61. Sadiq, S., Orlowska, M., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Inf. Syst. **30**(5), 349–378 (2005)
62. Schneider, K.: Verification of Reactive Systems: Formal Methods and Algorithms. Springer (2004)
63. Singh, M.P.: Semantical considerations on workflows: An algebra for intertask dependencies. In: Proc. of the 5th Int'l Workshop on Database Programming Languages, p. 5. Springer (1996)
64. The Business Rules Group: Defining business rules – what are they really? (2000). URL http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf
65. The Business Rules Group: The business motivation model - business governance in a volatile world. 1.2 (2007). URL http://www.businessrulesgroup.org/second_paper/BRG-BMM.pdf
66. Wagner, G.: How to design a general rule markup language. In: Proc. of the Workshop XML Technologies for the Semantic Web (2002)
67. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features – enhancing flexibility in process-aware information systems. Data & Knowledge Engineering **66**, 438–466 (2008)
68. Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S.: Providing integrated life cycle support in process-aware information systems. Int'l Journal of Cooperative Information Systems (IJCIS) Accepted for publication
69. Weber, I., Governatori, G., Hoffmann, J.: Approximate compliance checking for annotated process models. In: Proc. of the 1st Int'l Workshop on Governance, Risk and Compliance (GRCIS'08), pp. 46–60 (2008)
70. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proc. Semantics for BPM (2008)
71. Weske, M.: Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: HICSS-34, p. 7051 (2001)
72. Yu, J., Manh, T.P., Hand, J., Jin, Y.: Pattern-based property specification and verification for service composition. CeCSES Report SUT.CeCSES-TR010, Swinburne University of Technology (2006)