Compliance of Semantic Constraints – A Requirements Analysis for Process Management Systems*

Linh Thao Ly, Kevin Göser, Stefanie Rinderle-Ma, and Peter Dadam

Institute of Databases and Information Systems, University of Ulm, Germany
{thao.ly,kevin.goeser,stefanie.rinderle,peter.dadam}@uni-ulm.de

Abstract. Key to the use of process management systems (PrMS) in practice is their ability to facilitate the implementation, execution, and adaptation of business processes while still being able to ensure error-free process executions. Mechanisms have been developed to prevent errors at the *syntactic* level such as deadlocks. In many application domains, processes often have to comply with business level rules and policies (i.e., *semantic constraints*). Hence, in order to ensure error-free executions at the semantic level, PrMS need certain control mechanisms for validating and ensuring the compliance with semantic constraints throughout the process lifecycle. 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 meet the requirements and which challenges still have to be tackled. Finally, we show how the challenge of life time compliance can be dealt with by integrating design time and runtime process validation.

Key words: process management systems, semantic constraints, semantic process verification, compliance validation and enforcement

1 Introduction

Due to continuously changing market conditions, companies are forced to frequently adapt their business strategies in order to stay competitive [1–4]. Hence, there is a strong demand for process-aware information systems facilitating fast implementation and deployment of new business processes and allowing for flexible adaptations of existing ones. Process management systems (PrMS) are supposed to fulfill these demands and are therefore gaining increasing importance. Key to the application of PrMS technology in practice is their ability to allow for fast and flexible realisation of business processes on the one hand, while still being able to ensure error-free process executions on the other hand.

Much research efforts have been spent on avoiding errors at the *syntactic level* [5, 3, 6]. For example, by checking whether a process template (i.e., process

* This work was done within the research project "SeaFlows: Semantic Constraints in Process Management Systems", which is funded by the German Research Foundation (DFG), funding reference P5311001.

model) contains deadlocks or incorrect data links, a PrMS can guarantee for the absence of *syntactic* errors during process execution. Even if process instances have to be adapted at runtime in order to handle exceptional situations (e.g., by inserting additional activities), these checks [5] can be used for ensuring the syntactic correctness of process changes. These control mechanisms for process modeling and execution make PrMS an appealing development and execution environment for business processes.

Supporting solely checks at the syntactic level of processes, however, is not sufficient to ensure an 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) [7]. In the clinical domain, for example, clinical guidelines and pathways [8, 9] can be considered as example of such rules and policies. To clearly distinguish between syntactic constraints and business level rules and policies, we refer to the latter as *semantic constraints*[1]. For a particular business process, semantic constraints may express various dependencies such as ordering and temporal relations between activities, incompatibilities, and existence dependencies. For example, consider the following semantic constraints in natural language:

- *constraint* c_1*:* A patient should not be administered the drugs Aspirin and Marcumar within 5 days due to possible unwanted interactions
- *constraint* c_2: For patients older than 75, an additional tolerance test is required due to an increased risk
- *constraint* c_3: If an endoscopy and a gastroscopy are carried out for a patient within one week, the endoscopy will have to take place first due to possible interactions
- *constraint* c_4: 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 releasing

Obviously, such semantic constraints can be easily violated, in particular, if dynamic process changes are allowed during process execution (e.g., by dynamically inserting a gastroscopy before the endoscopy or by deleting the additional tolerance test for a patient with increased risk). Hence, there is an evident demand for control mechanisms which enable the PrMS to validate and to ensure the compliance of processes with semantic constraints.

Compliance validation has been addressed from various perspectives (e.g., compliance of cross-organisational workflows with business contracts, compliance of workflow transactions with predefined dependencies). Existing approaches either follow the paradigm of compliance validation at process template level (design time) or compliance monitoring at process instance level (runtime). However, we believe that PrMS have to provide more comprehensive support of semantic constraints. In particular, PrMS must be able to ensure the compliance over the complete process lifecycle (*life time compliance*).

[1] Semantic constraints can be considered a subset of *business rules* [10].

In previous work, we introduced a basic set of semantic constraints (i.e., binary exclusion and dependency constraints) expressing interdependencies between process activities [11]. Furthermore, we introduced an approach for validating processes and process changes against such constraints. In [12], this approach was extended in order to cope with concurrent changes. 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 (i.e., context-related) constraints. 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 PrMS 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 addition, we address the challenge of life time compliance in more detail. We advocate that life time compliance can only be achieved by providing an overall framework with adequate mechanisms for compliance and validation support in each phase of the process lifecycle.

This paper is structured as follows. Fundamental requirements are introduced in Sect. 2. In Sect. 3, state of the art is discussed. Our vision of an overall framework for life time compliance is presented in Sect. 4. Finally, a summary and an outlook on future research are provided in Sect. 5.

2 Fundamental Requirements for the Integrated Support of Semantic Constraints

Basically, for supporting semantic constraints in PrMS, existing PrMS concepts have to be supplemented by mechanisms for specifying semantic constraints and for assigning them to processes. Furthermore, mechanisms for validating and ensuring the compliance of processes with these semantic constraints have to be provided. From case studies (particularly of clinical processes, e.g. [13]) we derived fundamental requirements which have to be considered by a comprehensive approach. These fundamental requirements are discussed in the following.

2.1 Specifying and Integrating Constraints

Req. 1: A Formal Language for Constraint Specification 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 (e.g., to find out whether a constraint set is consistent, i.e., does not contain contradicting constraints). This, in turn, demands for a constraint specification language which has a formal foundation. In addition, the complexity of the specification language must not become an obstacle for constraint specification and for the

validation of processes against constraints. Thus, the main challenge is to find an appropriate balance between expressiveness, formal foundation, and potential analysis methods.

Req. 2: Constraint Organisation Although there are semantic constraints only relevant for one particular process (i.e., process specific), many semantic constraints (e.g., drug and therapy interactions) have a more global validity. An example of such a constraint with global validity is the ordering relation between the examinations endoscopy and gastroscopy. Hence, an appropriate way of organising semantic constraints (e.g., in a constraint repository [11] or a directory [7]) 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. Clinical guidelines, for example, may change due to new evidences in healthcare [8]. Since the lifecycle of constraints and the lifecycle of processes do not necessarily coincide, adequate mechanisms for versioning and propagating constraint changes to relevant processes have to be provided in order to support constraint evolution.

Req. 3: Support of Implementation independent and Implementation specific Constraints We identified two contradictory requirements regarding the abstraction level of constraint specification (cf. Fig. 1). On the one hand, a high level view on semantic constraints abstracting from implementation details has to be provided. The use of a non-technical level would allow for semantic constraints to be understood, managed, and specified by domain experts. Further, constraints (or what they refer to) may be implemented in various ways in a particular process. Thus, specifying semantic constraints at implementation level would restrict the possible (re)use of the constraints. 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 fit to the new process implementation even though its semantics has not changed. As example, consider again the constraints $\mathtt{c}_1$ and $\mathtt{c}_2$ from Sect. 1. Constraint $\mathtt{c}_1$ describes a drug interaction. For its semantics, it is irrelevant whether the drugs Aspirin and Marcumar are given to a patient within one process instance or within two separate process instances (e.g., when diagnostic procedures are mapped to different process models). For the semantics of constraint $\mathtt{c}_2$, 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 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.

The aforementioned examples show the demand for a high level view on semantic constraints. On the other hand, however, implementation level constraints are indispensable for process validation. For validating whether or not

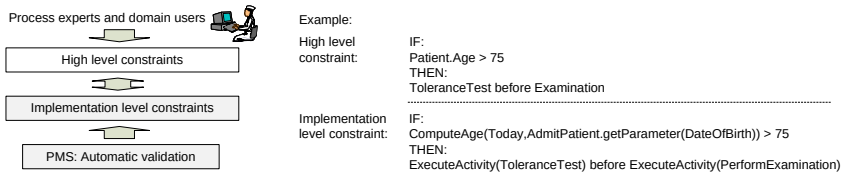

Fig. 1. Support of two abstraction levels for semantic constraints

an additional tolerance test for a patient is required in a particular process (according to c_2), for example, the PrMS has to know exactly how to determine the patient's age in this process (cf. Fig 1).

In summary, an approach for supporting semantic constraints in PrMS has to support a high level (i.e., conceptual) view on constraints focussing on their semantics as well as an implementation level view for constraint evaluation.

2.2 Ensuring Compliance

Req. 4: Support of Life Time Compliance Taking the lifecycle of processes in a PrMS into account, we identified four scenarios for semantic process validation:

◦ ***Req. 4.1: Compliance Validation at Design Time***
It is desirable to ensure the compliance with semantic constraints already at process template level (*compliance by design* [7]). A process template is described 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 template level, it is ensured that corresponding process instances are compliant as well. Consider process template P_1 in Fig. 2. P_1 only allows for process instances which are compliant with constraint c_2 (i.e., P_1 is compliant with c_2). For enforcing compliance at process template level, mechanisms have to be provided in order to validate process templates.

◦ ***Req. 4.2: Compliance Validation at Runtime***
Although being very essential, compliance by design is not always feasible for several reasons. As an example, consider a process which has to comply with a huge set of clinical guidelines with normative statements on what to do in exceptional cases (e.g., actions to take in case of a particular allergy). Enforcing

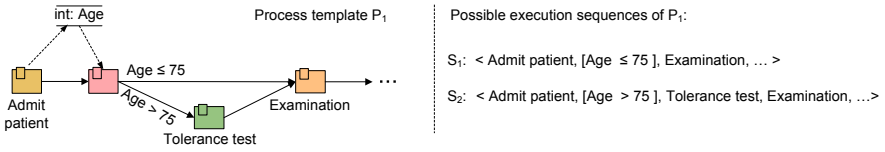

Fig. 2. A compliant process template

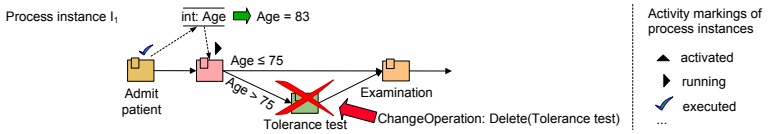

Fig. 3. A process instance change leading to semantic inconsistencies

the compliance with *all* these constraints at the process template level may lead to an overcomplex process template since each possible case described in a constraint has to be accounted for in the process structure of the template (e.g., by inserting corresponding conditional branches into the process template). Hence, depending on the nature of the constraints (e.g., how often an allergy occurs), it may be more feasible to postpone the enforcement of compliance with these constraints to 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 within 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 process template level without overcomplicating the process template.

◦ ***Req. 4.3: Validation of Process Changes***

Compliance validation also becomes necessary when process instances have to be modified during runtime in order to deal with exceptional situations [12]. In particular, if a process instance is frequently modified in an ad hoc manner by various agents with 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 $\mathtt{I}_1$ is modified by deleting the tolerance test (e.g., due to lack of time). This deletion, however, violates constraint $\mathtt{c}_2$ 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, an adaptive PrMS has to provide mechanisms to validate the compliance of process changes. Moreover, mechanisms to enforce compliance, for example, by refusing conflicting changes (e.g., refusing the deletion in Fig. 3) become essential as well. Since runtime validation often involves interaction with end users, efficient runtime checks are needed.

◦ ***Req. 4.4: Compliance Validation for Template Evolution***

When a process template is adapted (e.g., due to process optimization) it is often desirable that instances being executed according to the old template version also benefit from the changes at the template level [5] (*change propagation*). Since process instances may be individually modified, changes at the template level may be conflicting with changes at the instance level (e.g., administering two incompatible drugs at template and instance level) [12]. Hence, compliance checks for the propagation of template changes to process instances are essential to ensure life time compliance.

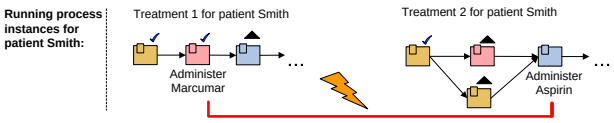

Fig. 4. Administering two incompatible drugs in two process instances

Req. 5: Support of Inter-Process Constraints Semantic constraints often affect more than only one process in a PrMS. As example, consider again constraint c_1. The incompatible drugs (Aspirin and Marcumar) may be given to a patient within *two* process instances (e.g., because the patient is undergoing two different treatments) (cf. Fig. 4). Hence, the support of semantic constraints must not be restricted to one process but has to be able to cross process boundaries.

Req. 6: Intelligible Feedback Semantic constraints are supposed to govern the process execution to ensure a semantically consistent procedure. 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 is required (e.g., reasons for the violation). In addition, feedback to help the user in finding adequate conflict avoidance (e.g., abstaining from a process change) and compensation strategies (e.g., inserting a compensation activity) is also essential (cf. 3.5).

Req. 7: Overrideable Constraints Semantic constraints are often not stringent. Many constraints are rather of recommendation nature [9] (i.e., soft constraints). Thus, constraint violations need not necessarily be an error. Therefore, it must be possible to override semantic constraints during process execution depending on the constraint's rigidity. Consider again constraint example c_1. A physician might still consider to administer both drugs. Prohibiting constraint overriding in such cases would annoy users and might even cause them to bypass the system.

Req. 8: Support of Traceability 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 [14].

3 State of the Art

In PrMS research, the main focus of approaches on process validation is to ensure the syntactic correctness of processes ([3–6]). Few approaches address constraints

other than of syntactic nature. In the following, a survey on existing approaches from PrMS research as well as related research areas which focus on ensuring the compliance of processes with constraints in a broader sense is provided. Existing approaches are first discussed with regard to the validation scenario they focus on (cf. Req. 4). A discussion with regard to 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 template) against certain constraints (cf. Req. 4.1). Existing approaches vary in constraint specification language, validation technique, and backgrounds.

In [15], an approach for achieving flexible processes is described which allows for the late modeling of subprocesses. Constraints expressing dependencies between activities are introduced for restricting composition possibilities. Before a subprocess is executed, it is validated against the constraints. In [16], an approach for compliance validation based on Concurrent Transaction Logic (CTR) is introduced. For validating a process (i.e., workflow graph) against constraints specified in CTR, the workflow graph is transformed into a CTR formula. This allows for the application of reasoning techniques for identifying semantic conflicts. Lu et al. [17] introduce an approach for measuring the compliance distance between a process template and a set of control objectives (comparable to constraints). The latter are specified in Formal Contract Language (FCL) [18]. Compliance is measured by comparing possible execution traces of the process model against ideal and sub-ideal execution traces.

In the context of web service composition, the question arises whether or not a choreography complies with certain constraints. In [19], Yu et al. introduce an approach for the specification of properties (i.e., constraints) and for validating BPEL processes against these properties. The properties are based on property patterns [20]. For process validation, a model checking approach is employed. Model checking has also been applied to process validation by several other approaches ([21, 22]). Foster et al. [23] introduce an approach for validating the interactions of web service compositions against obligation policies specified in the form of Message Sequence Charts. For validation, an approach based on Labeled Transition Systems is employed.

In [18], a priori 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 BPMN 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.

As discussed in Sect. 2.2, a priori validation is necessary for achieving compliance by design but also has its limitations. Since these approaches basically analyze the control flow (i.e., possible execution sequences), most dependencies involving abstraction levels more fine-grained than of process activities are not

within their scope. Though being suitable for certain scenarios, this level of abstraction is not appropriate for constraints involving context information (e.g., a patient's allergies or a customer's insurance sum) not expressed via activities.

3.2 Runtime Compliance Validation

The basic idea is to validate compliance by monitoring process-related events during runtime. Early approaches stem from rule-based transactions (e.g., [24, 25]). Their main focus is on scheduling upcoming process-related 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, Concurrent Temporal Logic) are used. In [26], an approach for specifying declarative process models using Linear Temporal Logic (LTL) is presented. For process enactment, the LTL formulas are synthesized into state automatons. In [27], 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 instance coordinator is used.

Monitoring runtime compliance has been addressed from the business contract perspective (e.g., [28–31]). In [29], process events are monitored to detect contract violations. In [30], contract clauses (constraints) 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. In [32], a similar approach is employed for monitoring the compliance of web service executions with choreographies.

In [33], a semantic mirror (i.e., a knowledge base of process events) 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 form of ECA rules. Agrawal et al. [34] also advocate the use of process monitoring for detecting non-compliance. In [35, 36], an approach for rule-based automatic instance adaptation is proposed. The rules are specified as ECA rules 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 action part of the rule.

Runtime compliance validation is particularly important for constraints involving runtime context information (cf. Req. 4.2). However, a limitation of most monitoring approaches is, that they do not allow for "look aheads". In particular, possible future process behavior is unknown. Decisions (e.g., enforcement decisions such as to reject a commit request) can only be made based on execution history so far. This leads to problems when constraints involve future behavior. In the scope of BPM, the information encoded in the process model (i.e., process structure) can be exploited at runtime in order to detect and avoid non-compliance in advance. We believe that a combination of both runtime and design time compliance checks is necessary for supporting life time compliance.

3.3 Check Point Metaphor

Business rule management systems (e.g., ILOG JRules [37]) 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 making points have to be predefined. Upon reaching the decision point at runtime, the rule service is invoked. In IBM Websphere, hard coded rules and checks are assigned to enforcement points where the rule service is invoked at runtime [38]. The check point paradigm is a complement to other validation scenarios. However, these approaches obviously cannot account for situations where more flexible checks are required (e.g., when processes are adapted).

3.4 A Posteriori Compliance Analysis

In [39], an approach for a posteriori validating processes against constraints is presented. Constraints specified in Linear Temporal Logic are verified over process logs. This approach is not applicable in scenarios where 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 Discussion and Summary

Though the expressiveness varies, common to most approaches is a certain degree of formal foundation of the specification language (cf. Req. 1). The requirement of high level and implementation level constraints (Req. 3) has been addressed in [33]. 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 are potentially able to deal with inter-process constraints (Req. 5) by nature. To evaluate to what extent existing 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, Req. 4.3, Req. 4.4). In [33], recovery strategies for control violations are proposed (e.g., rollback or ignoring the violation). This is an interesting approach which treats constraint violations like process exceptions in general. However, these recovery strategies are applied after a constraint is already violated. In our opinion, strategies for avoiding a violation are required as well. In [40], an approach for auditing (BPMN) process models for compliance by annotating activities with effects is introduced. For compliance resolution compliance patterns are proposed.

As discussed, existing approaches either focus on validating process templates at design time or on compliance monitoring at runtime. The validation of process changes has not been addressed (Req. 4.3 and Req. 4.4). 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 ensuring

compliance over the complete process lifecycle. Our objective in the SeaFlows project is to provide a fundamental framework which allows for comprehensive support of semantic constraints in adaptive PrMS. In this context, mechanisms for ensuring life time compliance are essential.

4 Towards Life Time Compliance – A Vision

As discussed, many challenges still have to be tackled in order to achieve comprehensive support of semantic constraints in PrMS. In the following, we address the particular challenge of ensuring life time compliance (Req. 4).

For achieving compliance throughout the complete process lifecycle, adequate mechanisms for supporting and ensuring compliance in each phase of the process lifecycle are required. Our vision of the key mechanisms is sketched in Fig. 5. In the following, we explain the particular compliance support envisaged for each phase. Due to space limitations, we abstain from presenting our ideas on change propagation (Req. 4.4).

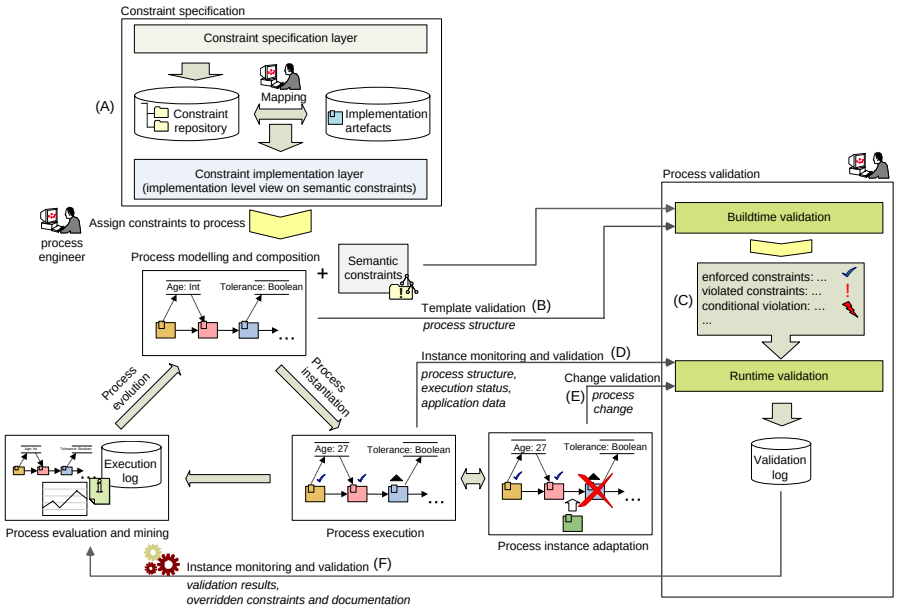

Fig. 5. Key mechanisms for life time compliance

4.1 Design Time: Process Modeling and Composition

Constraint Specification To support a high level view on semantic constraints as well as an implementation level view for automatic validation (Req. 3), an

overall framework has to provide two layers (cf. Fig. 5 (A)). We envisage a *constraint specification layer* providing mechanisms for constraint specification, analysis, and management at an implementation independent level. Many approaches for constraint specification have been proposed in literature (e.g., LTL, FCL, or the SBVR standard [41]). To evaluate them and to find a suitable constraint language, a detailed analysis of semantic constraints is vital.

To obtain implementation level constraints (*constraint implementation layer*), concepts used in semantic constraints (i.e., constraint artifacts) are mapped to corresponding implementation artifacts (e.g., process activities, subprocesses, or process data) by a process engineer (cf. Fig. 5 (A)). The decoupling of constraint semantics and constraint implementation allows for changes at the implementation level as well as changes at the specification level without affecting each other (Req. 3).

Semantic constraints may be stored in a constraint repository and assigned to categories for facilitating constraint reuse. To assign semantic constraints to a process, the constraint repository may be browsed for existing constraint sets (e.g., drug interactions) or new ones may be created (cf. Fig. 5).

Process Template Validation Following Req. 4.1, mechanisms for template validation already during design time are envisaged (cf. Fig. 5 (B)). At this stage, only the process structure may serve as input for compliance validation. However, many semantic constraints involve runtime information (e.g., data conditions) which are not available at design time. Hence, in order to provide the process engineer with helpful validation results we envisage fine-grained notions of constraint violations (e.g., whether a constraint will be violated in all possible process instances of the template, or whether it will only be violated in a process instance under particular conditions, cf. Fig. 5 (C)). Such fine-grained notions of constraint violations would allow for more fine-grained validation results and feedback, which, in turn, can help the process engineer to evaluate and to enhance the process template.

4.2 Runtime: Process Execution and Process Instance Adaptation

It is not always feasible to enforce all semantic constraints at the process template level (cf. Req. 4.2). Hence, it must be possible to instantiate process instances from a process template 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 data values). The evaluation and validation of corresponding constraints based on the runtime information has to be carried out during process execution (cf. Fig. 5 (D)). Our objective is to identify *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 behavior of the instance (i.e., no mere monitoring). How this can be tackled is part of ongoing research.

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 template level and thus, do not require costly compliance checks at runtime. 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 mechanisms for validating process changes (Req. 4.3), corresponding compliance checks have to be integrated into existing process adaptation mechanisms of PrMS (cf. Fig. 5 (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 be exploited. In [12], we developed an approach for evaluating only semantic constraints which might be violated by the particular process change. So far, this approach is restricted to activity-level constraints and is to 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 may provide meaningful input for process mining (cf. Fig. 5 (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 Summary and Outlook

The demand for process compliance with rules and policies leads to new requirements on PrMS technology. The support of semantic constraints would leverage the applicability of PrMS technology in practice since not only the syntactic level of processes can be supported by PrMS. Processes may also be enriched with semantic constraints for ensuring semantically consistent process executions. In this paper, we identified fundamental requirements for supporting semantic constraints in PrMS. Furthermore, we provided a survey on existing approaches and discussed to what extent they are able to meet the requirements. We showed that there is an demand for an approach which allows for supporting the compliance

over the complete process lifecycle. In addition, we presented our vision of the key mechanisms of a framework for realising life time compliance.

As we also pointed out in this paper, there are many open questions to be analyzed in more detail in order to develop a truly comprehensive approach. Part of ongoing work in the SeaFlows project is a detailed analysis of semantic constraints. This would allow for an evaluation of existing constraint specification languages. Furthermore, a closer look at the relations between constraint types and appropriate validation scenarios is necessary for further optimizing the interplay between design time and runtime validation mechanisms.

References

1. van der Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoret. Comp. Science **270** (2002) 125–203
2. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. DKE **24** (1998) 211–238
3. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases **16** (2004) 91–116
4. Weske, M.: Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In: HICSS-34. (2001) 7051
5. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – A survey. DKE **50** (2004) 9–34
6. van der Aalst, W.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: Proc. BPM '00. (2000) 161–183
7. Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: Proc. BPM '07. (2007) 149–164
8. Newcastle Guideline Development and Research Unit: Management of dyspepsia in adults in primary care. (2004)
9. Lenz, R., Reichert, M.: It support for healthcare processes - premises, challenges, perspectives. Data Knowl. Eng. **61** (2007) 39–58
10. Wagner, G.: How to design a general rule markup language. In: Proc. of the Workshop XML Technologies for the Semantic Web. (2002) Interessante Übersicht zum Thema Spezifikation und andere Aspekte von Regeln in verschiedenen Bereichen wie SQL, OCL, Prolog, Business Rules etc.
11. Ly, L.T., Rinderle, S., Dadam, P.: Semantic correctness in adaptive process management systems. In: Proc. BPM '06. Volume 4102 of LNCS. (2006) 193–208
12. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. DKE (**64**) 3–23
13. Konyen, I., Reichert, M., Schultheiß, B., Frank, S., Mangold, R.: Process design for minimally-invasive surgery. Interne Ulmer Informatik-Berichte DBIS-14, University of Ulm, DBIS Institute (1996) In german.
14. Peleg, M., Soffer, P., Ghattas, J.: Mining process execution and outcomes. In: 1st Int. Workshop on Process-oriented Information Systems in Healthcare. (2007)
15. Sadiq, S., Orlowska, M., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Inf. Syst. **30** (2005) 349–378
16. Davulcu, H., Kifer, M., Ramakrishnan, C.R., Ramakrishnan, I.V.: Logic based modeling and analysis of workflows. In: PODS '98. (1998) 25–33
17. Lu, R., Sadiq, S., Governatori, G.: Compliance aware process design. In: Proc. BPM Workshops '07. (2007)

18. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Proc. EDOC '06. (2006) 221–232
19. 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)
20. Dwyer, M., Avrunin, G., Corbett, J.: Patterns in property specifications for finite-state verification. In: Proc. 21st Int. Conf. on Software Engineering. (1999) 411–420
21. Fötsch, D., Pulvermüller, E., Rossak, W.: Modeling and verifying workflow-based regulations. In: Proc. of Workshop on Regulations Modelling and their Validation and Verification. (2006) 825–830
22. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal **46** (2007) 335–261
23. Foster, H., Uchitel, S., Magee, J., Kramer, J.: Model-based analysis of obligations in web service choreography. In: Proc. AICT-ICIW '06. (2006)
24. Attie, P., Singh, M., Sheth, A., Rusinkiewicz, M.: Specifying and enforcing intertask dependencies. In: Proc. VLDB '93. (1993) 134–145
25. Singh, M.: Semantical considerations on workflows: An algebra for intertask dependencies. In: Proc. 5th Int. Workshop on Database Programming Languages, London, UK, Springer-Verlag (1996) 5
26. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Proc. BPM Workshops '06. (2006) 169–180
27. Heinlein, C.: Workflow and process synchronisation with interaction expressions and graphs. In: Proc. ICDE '01. (2001) 243–252
28. van den Heuvel, J., Weigand, H.: Cross-organizational workflow integration using contracts. In: Proc. Business Object Workshop '00. (2000)
29. Milosevic, Z., Josang, A., Dimitrakos, T., Patton, M.: Discretionary enforcement of electronic contracts. In: Proc. EDOC '02. (2002) 39–50
30. Alberti, M., et al.: Expressing and verifying business contracts with abductive logic programming. In: Normative Multi-agent Systems. (2007)
31. Giblin, C., Müller, S., Pfitzmann, B.: From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Technical report, IBM (2006)
32. Alberti, M., et al.: Computational logic for run-time verification of web services choreographies: Exploiting the SOCS-SI tool. In: WS-FM. (2006) 58–72
33. Namiri, K., Stojanovic, N.: Pattern-based design and validation of business process compliance. In: OTM Conferences (1). (2007)
34. Agrawal, R., Johnson, C., Kiernan, J., Leymann, F.: Taming compliance with Sarbanes-Oxley internal controls using database technology. In: Proc. ICDE '06. (2006) 92–92
35. Greiner, U., et al.: Adaptive guideline-based treatment workflows with AdaptFlow. In: Proc. CGP '04. Volume 101. (2004) 113–117
36. Müller, R., Greiner, U., Rahm, E.: Agentwork: A workflow system supporting rule-based workflow adaption. Data & Knowledge Engineering **51** (2004) 223–256
37. ILOG: ILOG JRules and IBM MQWF – White Paper. (2005)
38. Goldszmidt, G., Joseph, J., Sachdeva, N.: On demand business process life cycle, part 6: Apply customization policies and rules. Technical report, IBM (2005)
39. van der Aalst, W., de Beer, H., van Dongen, B.: Process mining and verification of properties: An approach based on temporal logic. In: Proc. CoopIS '05. (2005)
40. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Proc. ICSOC '07. Volume 4749 of LNCS. (2007) 169–180
41. The Business Rules Team: Semantics of business vocabulary & business rules. In: W3C Workshop on Rule Languages for Interoperability. (2005)