SeaFlows Toolset – Compliance Verification Made Easy for Process-Aware Information Systems*

Linh Thao Ly[1], David Knuplesch[1], Stefanie Rinderle-Ma[2], Kevin Göser[3], Holger Pfeifer[4], Manfred Reichert[1], and Peter Dadam[1]

[1] Institute of Databases and Information Systems
Ulm University, Germany
{thao.ly,david.knuplesch,manfred.reichert,peter.dadam}@uni-ulm.de
[2] Institute of Artificial Intelligence
Ulm University, Germany
holger.pfeifer@uni-ulm.de
[3] Faculty of Computer Science
University of Vienna, Austria
stefanie.rinderle-ma@univie.ac.at
[4] AristaFlow GmbH, Germany
kevin.goeser@aristaflow.com

Abstract. In the light of an increasing demand on business process compliance, the verification of process models against compliance rules has become essential in enterprise computing. The SeaFlows Toolset featured in this paper extends process-aware information systems with compliance checking functionality. It provides a user-friendly environment for modeling compliance rules using a graph-based formalism and for enriching process models with these rules. To address a multitude of verification settings, we provide two complementary compliance checking approaches: The *structural compliance checking* approach derives structural criteria from compliance rules and applies them to detect incompliance. The *data-aware behavioral compliance checking* approach addresses the state explosion problem that can occur when the data dimension is explored during compliance checking. It performs context-sensitive automatic abstraction to derive an abstract process model which is more compact with regard to the data dimension enabling more efficient compliance checking. Altogether, SeaFlows Toolset constitutes a comprehensive and extensible framework for compliance checking of process models.

Keywords: Compliance rules, Data-aware compliance checking, Process verification.

1 Introduction

In the light of an increasing demand for business process compliance [1], the verification of process models within process-aware information systems against

* This work was done in the research project SeaFlows which is partially funded by the German Research Foundation (DFG).

P. Soffer and E. Proper (Eds.): CAiSE Forum 2010, LNBIP 72, pp. 76–91, 2010.

compliance rules has become essential in enterprise computing. To ensure compliance with imposed rules and policies, compliance audits for process models become necessary. Due to the increasing complexity of process models [2], manual compliance verification is hardly feasible. Tool support is particularly needed in order to deal with changes at different levels. On the one hand, changes of regulatories and policies occur which then necessitate leading to changes of implemented compliance rules. On the other hand, changes to business processes may take place, which result in changes of implemented process models. This further necessitates tool support for (semi-)automatic compliance verification.

In this paper, we introduce SeaFlows Toolset, a tool framework for business process compliance verification, and underlying concepts. These resulted from our research in the SeaFlows project. In this project, we aim at providing techniques to enable compliance with imposed regulatories throughout the process lifecycle. This includes compliance checking of business process models at buildtime, but also requires compliance monitoring for process instances at runtime [3]. With the implementation of SeaFlows Toolset, so far, we have realized concepts addressing compliance checking of process models at buildtime. In particular, the toolset enables to visually model compliance rules by means of so-called *compliance rule graphs*. In addition, it supports the verification of process models against imposed compliance rules. To support a variety of verification scenarios and to exploit their specific properties, we introduce two complementary verification approaches. First, we discuss a *structural compliance checking* approach based on node relations, which enables efficient compliance verification for block-structured process models. In addition, we provide a general *behavioral compliance checking* approach that realizes data-awareness as well.

Example. Throughout this paper an exemplary order-to-delivery scenario will be used to illustrate basic concepts and SeaFlows Toolset: Fig. 1 depicts a process model P for a simplified order-to-delivery process. For brevity, we abstain from modeling the complete data flows of P. The order-to-delivery process, in turn, may be subject to the compliance rules collected in Table 1. They constrain the execution and ordering of activities and events within a process model.

The remainder of this paper is structured as follows. In the following, we first introduce the concepts behind SeaFlows Toolset before presenting it in more detail. Compliance rule graphs, the visual modeling formalism, are introduced in Sect. 2. The structural as well as the behavioral compliance checking approach are discussed in Sect. 3. The particular components of SeaFlows Toolset are introduced in Sect. 4. Related work is discussed in Sect. 5 before we close the paper with an outlook on future developments in Sect. 6.

2 Compliance Rule Graphs – A Visual Modeling Formalism

Prerequisite to automatic compliance checking is that compliance rules are modeled using a suitable formalism. On the one hand, the formalism has to be sufficiently expressive. On the other hand, the its complexity should not impede

Table 1. Examples of compliance rules for order-to-delivery processes

c_1	A received order has to be either confirmed or declined.
c_2	Outsourced production shall be followed by a quality test.
c_3	After confirming an order and previous to confirming shipping, shipping has to be prepared and eventually shipped.
c_4	Premium customer status shall only be offered after a prior solvency check.
c_5	For orders of a non-premium customer with a piece number beyond `80,000`, a solvency check becomes necessary before assessing the order.
c_6	After confirming an order of a non-premium customer with piece number of at least `125,000`, premium status should be offered to the customer
c_7	Shipping orders with piece number below `80,000` does not require shipping insurance.
c_8	Orders with piece number beyond 40,000 shall only be confirmed after prior approval.

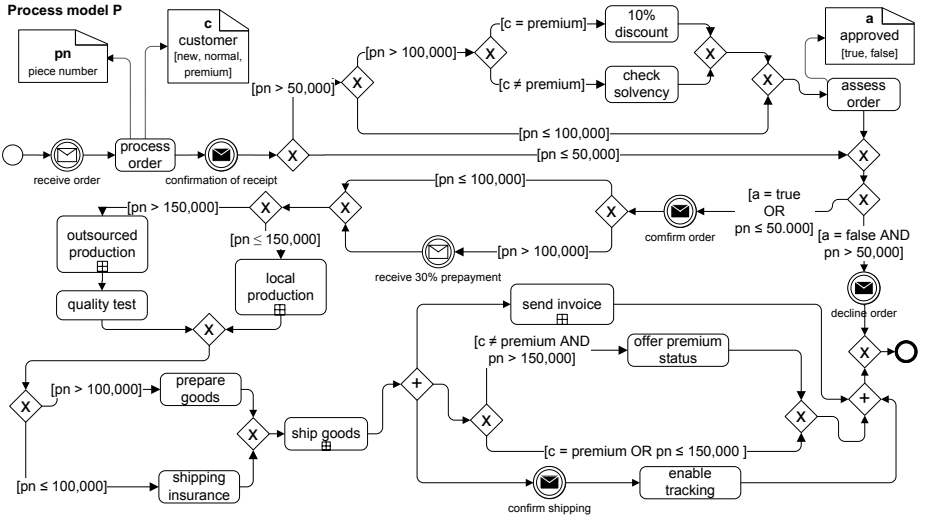

Fig. 1. An order-to-delivery process (modeled with BPMN)

its application. Apparently, logic formalisms, such as linear temporal logic, are powerful. However, their complexity can become a barrier to their application to compliance rule modeling by domain experts in practice. To address this issue, pattern-based approaches [4, 5, 6] have been suggested recently (cf. Sect. 5). Visual patterns are provided which represent placeholders for logic formulas. However, pattern-based approaches provide only limited support for modeling more complex compliance rules. Hence, we opted for a compositional graph-based

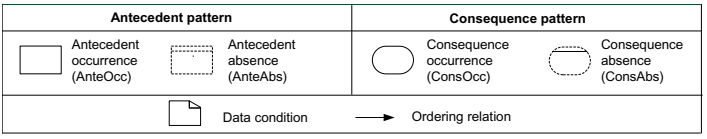

Fig. 2. "Ingredients" for modeling compliance rule graphs

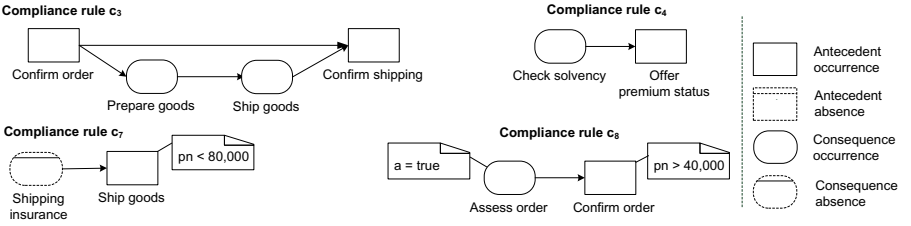

Fig. 3. Compliance rule graphs

approach that provides both a visual formalism and formal semantics. The basic idea is to enable compliance rule modeling in a way similar to process modeling, namely by means of graphs representing fragments of process executions.

In our case studies (e.g., in the clinical domain) we identified the typical structure of compliance rules. Roughly described, it states that if some event patterns occur then some other event patterns must also occur. Each event pattern, in turn, can express the occurrence or absence of particular events, or be structured in a complex manner (e.g., particular relations between events). Based on these observations, we developed our compliance rule graph (CRG) approach that enables the modeling of compliance rules by means of directed graphs. The basic "ingredients" for composing a CRG are depicted in Fig. 2. They comprise four different node types indicating occurrence and absence of activities of an associated type. AnteOcc and AnteAbs are dedicated to modeling the antecedent pattern activating a compliance rule whereas ConsOcc and ConsAbs are dedicated to modeling consequence patterns that have to occur upon activation of a compliance rule. CRG nodes can be related to each other using ordering relations. In addition, a CRG can be enriched with data conditions to be able to capture data-aware compliance rules. Fig. 3 shows how these "ingredients" are applied to model some of the compliance rules from Table 1. CRG c_3, for example, states that if the antecedent pattern consisting of the sequence `confirm order` and `confirm shipping` occurs in a process execution, the sequence consisting of `prepare goods` and `ship goods` must occur in between. CRG c_8, in turn, shows how data conditions are applied. It expresses that activity `assess order` with data condition `a = true` is required prior to confirming an order with data condition `pn` $>$ `40,000`.

CRGs go beyond a pure visual notation. We also equipped them with formal semantics that enable formal compliance verification. Due to space limitations,

we omit the definition of CRG formal semantics and refer to [7], where CRG formulas and their interpretation over execution traces are discussed.

3 Structural and Behavioral Compliance Checking

Constraining process behavior, compliance rules typically are specified at a behavioral level whereas a process model is a structure describing possible process behavior. To verify process models against such behavioral compliance rules, we basically have two options:

Behavioral Compliance Checking is conducted by verifying the process behaviour against imposed compliance rules. This can be achieved by exploring possible process behavior with regard to compliance.

Structural Compliance Checking derives *structural properties* from behavioral compliance rules. Such structural properties can be applied to check the process model for compliance with imposed rules. Depending on the process meta model employed, structural compliance checking can be conducted in a more efficient manner than behavioral compliance checking.

We can draw parallels to Petri Nets research. Since reachability analysis is costly, researchers also developed strategies to structurally analyze Petri Nets which promise a more efficient checking of certain Petri Net properties [8].

While behavioral compliance checking is a general approach and thus is broadly applicable, efficient structural compliance checking relies on certain conditions. For example, structural compliance checking of data-aware compliance rules is rather not feasible, since the data dimension has to be explored. To provide support for a multitude of compliance verification settings, we integrated a structural as well as a behavioral compliance checking approach into SeaFlows Toolset. The structural compliance checking approach, introduced in Sect. 3.1, conducts efficient compliance checking for a subset of CRGs. The behavioral approach, discussed in Sect. 3.2, addresses data-aware compliance checking and provides strategies to avoid state explosion.

3.1 Structural Compliance Checking Based on Node Relations

The basic idea of our structural compliance checking approach is to automatically derive *structural criteria* on the process model from behavioral compliance rules. We first introduced structural compliance checking in [9], however, so far we only addressed basic exclusion and dependency constraints. We have further extended our approach to provide support for a broader range of compliance rules. Based on the assumption of unique labels (i.e., unique occurrences of activities within a process model), we have developed an efficient structural compliance checking approach for a subset of CRGs. This approach is designed to support loop-free process models and abstracts from data conditions.

Our structural compliance checking approach is conducted in three steps as illustrated in Fig. 4. In Step 1, for each CRG, a set of structural criteria to be

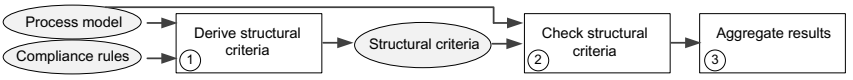

Fig. 4. The process of structural compliance checking

checked over the process model is determined automatically. These criteria can be considered queries on the relations of nodes within the process model (i.e., *node relations*) that are relevant to the compliance rule. We define five structural criteria consisting of containment, occurrence, and precedence relations:

$\oplus$ **A (contains A)** is a unary structural containment relation that applies if A is contained in the process model.

A $\otimes$ B (A excludes B) is a structural occurrence relation that applies if A and B are located on different branches of an exclusive gateway. For example, `10% discount` $\otimes$ `check solvency` applies to process model P from Fig. 1.

A $\triangleright$ B (A implies B) is a non-directed structural occurrence relation. In order for it to evaluate to `true`, B must not be located on a branch of an exclusive gateway, on which A is not located as well, provided that A and B are both present in the process model. For example, `assess order` $\triangleright$ `check solvency` will be evaluated to `false` over the order-to-delivery process from Fig. 1, since `check solvency` is located on a branch of an exclusive gateway (i.e., branch with condition `c=premium`) while `assess order` is not located on that branch. This means, that `check solvency` will be executed optionally to `assess order`. However, the node relation `check solvency` $\triangleright$ `assess order` will be evaluated to `true`, since `assess order` is located on the exclusive branch with data condition `pn>50,000` and `check solvency` is also located on this branch.

A $\triangleright$ $B_1|B_2|\ldots|B_n$ (A implies B_1,B_2,..., B_n) is a non-directed structural occurrence relation. It applies if A is always executed together with B_1,B_2,..., or B_n. At the structural level, this criterion is checked by adopting strategies from data flow analysis.

A $\gg$ B (A precedes B) is a structural precedence relation that applies if there is a directed path in the process model leading from A to B. For example, `prepare goods` $\gg$ `ship goods` will be evaluated to `true` over the order-to-delivery process from Fig. 1.

In Step 2, the process model is checked for compliance with the derived structural criteria. Thus, we can precisely identify those structural criteria causing incompliance. In case a compliance violation is detected, these structural criteria will be collected in Step 3 and will be used for error diagnosis and for the generation of intelligible feedback. By showing the process designer, which structural criteria are not satisfied by the process model, the system can help to resolve incompliance. By exploiting properties of the process meta model properties such as block-structuring, the structural criteria can be evaluated efficiently. Adopting the paradigm of dynamic programming, we cache node relations already queried to enable faster evaluation if the same relations are queried a second time.

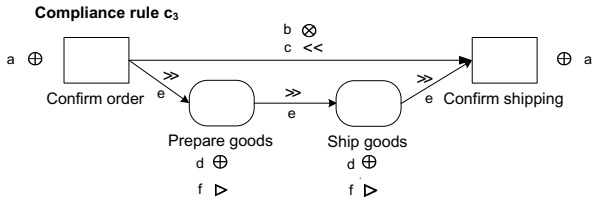

Fig. 5. A CRG annotated with structural criteria to be applied for conducting structural compliance checking

In the following, we will sketch how structural criteria are derived and checked for compliance rule c_3 from Table 1 and the corresponding CRG from Fig. 3.

Example. In Step 1, structural criteria to be queried in order to verify compliance with c_3 are derived based on c_3's structure. Fig. 5 depicts CRG c_3 annotated with types of structural criteria that will be applied during structural compliance checking and the order of application. Exploiting the antecedent-consequence structure of CRGs, it is first checked whether the process behavior modeled by the CRG's antecedent pattern can be produced by the process model. For CRG c_3 the antecedent pattern consists of `confirm order`, `confirm shipping`, and the ordering relation between them. If the antecedent can be activated (i.e., the pattern can be produced by the process model), structural compliance checking proceeds with checking compliance with the consequence pattern.

a. Premise to the activation of an antecedent pattern by a process model is that relevant antecedent activities are contained in the model. Hence, as the first step, it is checked wether the activities `confirm order` and `confirm shipping` assigned to the AnteOcc nodes are both contained in the process model using the $\oplus$ relation. If one of them is not contained in the process model, the process behavior modeled by the antecedent CRG can never be produced and further checking would not be necessary.

b. The antecedent pattern can only be produced by a process model if the AnteOcc activities `confirm order` and `confirm shipping` are located in the model such that they can be executed together. Whether this applies can be determined by checking whether these activities exclude each other by querying `confirm order` $\otimes$ `confirm shipping`. If this is the case, the antecedent pattern can never occur, which, in turn, makes further compliance checking superfluous.

c. The AnteOcc activities `confirm order` and `confirm shipping` are assigned an ordering relation in c_3. To check whether ordering relations apply, the $\gg$ relation is employed. Provided that a and b have not led to an interruption of the checking, `confirm order` can only be followed by `comfirm shipping` if precedence relation `confirm shipping` $\gg$ `confirm order` does not apply (i.e., the activities are ordered the other way around or in parallel branches).

d. In order to check compliance with the consequence pattern, it is first queried whether the relevant activities `prepare goods` and `ship goods` are contained in the process model. This is done by using the $\oplus$ relation. If one of these acitivites

are not contained in P, the consequence of the compliance rule apparently will never be satisfied and compliance checking can be ceased.

e. Ordering relations modeled in a CRG's consequence pattern also have to be verified. In our example, the correct ordering of `prepare goods` and `ship goods` with regard to each other as well as with regard to the AnteOcc activities `confirm order`and `confirm shipping` is checked by applying the $\gg$ relation.

f. According to c_3, `prepare goods` and `ship goods` always have to be executed when `confirm order` and `confirm shipping` are executed. This can be queried using the $\triangleright$ relation. Activity `prepare goods` for example will always be executed when `confirm order` and `confirm shipping` are executed, if it is structurally implied by one of the latter activities. This can be checked by apply the following query: (`confirm order` $\triangleright$ `ship goods`) $\vee$ (`confirm shipping` $\triangleright$ `ship goods`). If the query applies, the execution of `ship goods` together with `confirm order` and `confirm shipping` will be ensured. The same has to be checked for `prepare goods`.

Fig. 6 shows the results of querying the derived structural criteria over the order-to-delivery process P from Fig. 1 as obtained in Step 2. The criteria are arranged along a decision tree. Particular query results are printed in boldface. As Fig. 6 shows, the antecedent pattern can be produced by P. In addition, the required activities of the consequence pattern (i.e., `prepare goods` and `ship goods`) are both contained in P and fulfill the required precedence relations. However, while occurrence relation query OR3 is evaluated to `true`, OR2 does not

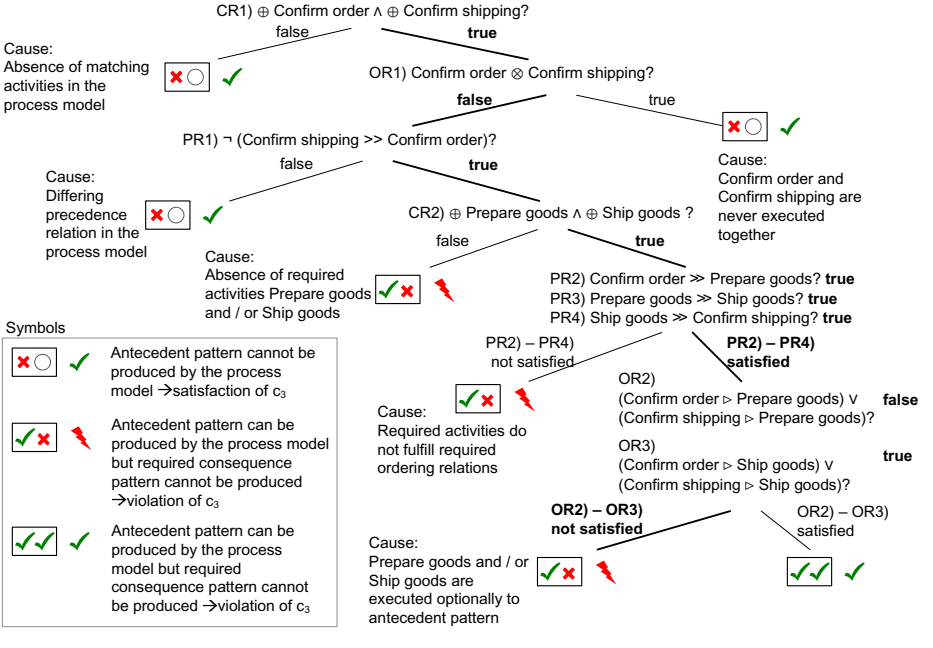

Fig. 6. Structural compliance checking of c_3 against the order-to-delivery process

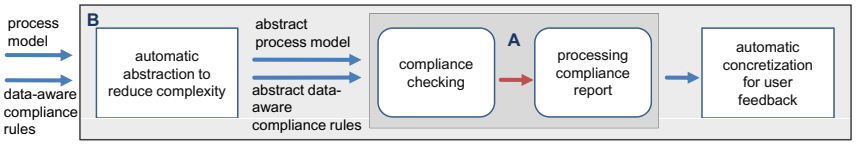

Fig. 7. Abstraction and concretization as pre- and postprocessing steps to the actual data-aware compliance checking

apply. This means that `prepare goods` is executed optionally to both `confirm order` and `confirm shipping`. Thus, at the behavioral level, executing P can lead to execution traces containing both `confirm order` and `confirm shipping` but not `prepare goods`. Hence, P does not comply with c_3.

In Step 3, the results obtained from querying the structural criteria can be aggregated and used to generate error diagnosis. In our example, we can precisely identify the violation of OR2 as the cause of incompliance. In Sect. 4.2, we will show how the *SeaFlows Structural Compliance Checker* uses the query results to generate compliance reports.

3.2 Data-Aware Behavioral Compliance Checking

As previously discussed, behavioral compliance checking is a general approach, and thus, broadly applicable. Its particular advantage over structural compliance checking becomes manifest, when it comes to data-aware compliance checking. Data-awareness is vital to support a variety of compliance verification settings. A closer look at compliance rules and scenarios reveals two major scenarios:

a) Compliance rules without data conditions, but the process model contains data-based gateways that can affect the verification outcome.
b) Compliance rules containing data conditions.

An example of case a) is given by compliance rule c_4 from Table 1. While c_4 does not contain any data conditions, abstracting from the data perspective when checking compliance of the order-to-delivery process P from Fig. 1 with c_4 would lead to an erroneous compliance report. In particular, abstracting from the data-based gateways contained in process model P would lead to the conclusion that a solvency check is not sure to be executed prior to offering premium customer status. However, due to the correlation of data-based gateways within P, it is ensured that premium customer status is only offered after a prior solvency check. Note that `check solvency` is carried out for non-premium customers and `pn>100,000` while `offer premium status` is only carried out for non-premium customers and `pn>150,000`. Hence, the order-to-delivery process, in fact, complies with c_4. For examples of case b), consider compliance rules c_7 and c_8 from Table 1 and corresponding CRGs in Fig. 3, each containing data conditions.

As these examples show, in both cases, data-awareness is necessary to provide correct compliance reports. The challenge with data-aware compliance checking

is that the exploration of the data dimension during compliance checking can lead to state explosion and thus, to intractable complexity. To tackle this, we introduced abstraction strategies to reduce the complexity of data-aware compliance checking [10]. By analyzing the data conditions contained in the compliance rule and in the process model, our approach reduces the state space of the data dimension to be explored during verification. This is achieved by abstracting from concrete states of data objects to abstract states. Based on the compliance rules to be checked our approach *automatically* derives an *abstract process model* and corresponding *abstract compliance rules* (cf. Fig. 7 B). The abstract process model is more compact than the original process model with regard to the data dimension. Thus, it enables more efficient exploration of the data dimension when being used as input to the actual compliance checking (cf. Fig. 7 A).

For conducting behavioral compliance checking for the abstract process model, we apply model checking techniques. In case of violation, the counterexample obtained from the model checker is concretized to yield not only the incompliant execution but also its data conditions. To address both scenarios a) and b), the *SeaFlows Data-aware Compliance Checker* is able to deal with data-aware compliance rules (i.e., compliance rules containing data conditions) and data conditions in process control flow (i.e., data-based gateways).

4 SeaFlows Toolset

We implemented the concepts introduced here in SeaFlows Toolset. It extends process-aware information systems (PAIS) by compliance checking functionality. Fig. 8 depicts the interplay between existing infrastructure stemming from PAIS (e.g., activity repository, process modeling tool, and process model repository) and components introduced by SeaFlows Toolset[1].

The *SeaFlows Graphical Compliance Rule Editor* (cf. Fig. 8) allows to model CRGs over process artifacts (cf. Sect. 2). By interacting with the activity repository managing process artifacts relevant within a business domain, the Graphical Compliance Rule Editor enables compliance rule modeling over exactly the process artifacts available in the domain. Thus, we can enrich process models by compliance rules that are imposed on the corresponding business process. This can be done at an early stage, when the process is modeled to enable *compliance by design*. Compliance rules may be also assigned to a completed or released process model to conduct compliance audits.

SeaFlows Toolset currently comprises two compliance checking components to verify process models (cf. Fig. 8). The *Structural Compliance Checker* implements the structural compliance checking approach while the *Data-aware Compliance Checker* employs a behavioral approach and addresses data-awareness. By interacting with the process modeling tool of PAIS, the SeaFlows compliance checkers enable process designers to verify process models already during process

[1] The Rule Graph Execution Engine for executing CRGs is currently under implementation.

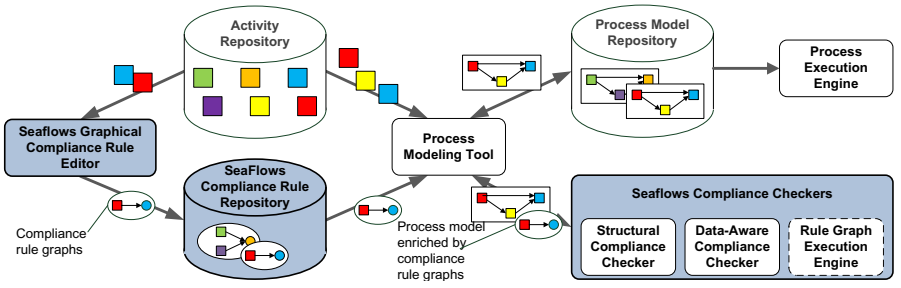

Fig. 8. Overall infrastructure around the SeaFlows Toolset

design. Meaningful compliance reports help process designers to identify incompliant process behavior. Based on them, process designers may further modify the process model until incompliance is resolved.

To transfer our concepts into a comprehensive prototype, we opted to base our implementation on AristaFlow BPM Suite which originated from research activities in the ADEPT project [11]. AristaFlow BPM Suite provides a powerful application programming interface (API) which enables us to extend existing PAIS functionality by compliance checking mechanisms in an elegant manner. Thus, SeaFlows compliance checking components are smoothly integrated into the process modeling environment of AristaFlow BPM Suite. In the following, the components of SeaFlows Toolset are discussed in more detail.

4.1 Graphical Compliance Rule Editor and Compliance Rule Repository

The *Graphical Compliance Rule Editor* provides a user-friendly environment for modeling CRGs. Fig. 9 shows c_3, c_4, c_7, and c_8 as modeled using the compliance rule editor. Nodes of CRGs are assigned to activity types available in the activity repository (cf. Fig. 8). Modeled CRGs are exported as separate XML-files which enables their organization within rule sets in the Compliance Rule Repository. In addition, versioning and collaborative modeling of compliance rules are also supported by the repository. Being implemented based on the Eclipse Modeling Framework, modeled CRGs are based on a defined data object model which facilitates their import and processing in compliance checking tools.

4.2 Structural Compliance Checker

Based on the results of checking the structural criteria, the *Structural Compliance Checker* is able to provide detailed diagnosis helpful to locate incompliance. Fig. 10 shows the compliance report for checking compliance rule c_3 against the order-to-delivery process P. Corresponding to Fig. 6, the Structural Compliance Checker identifies that while there is no problem with required activity `ship goods`, the other required activity `prepare goods` is optional to both

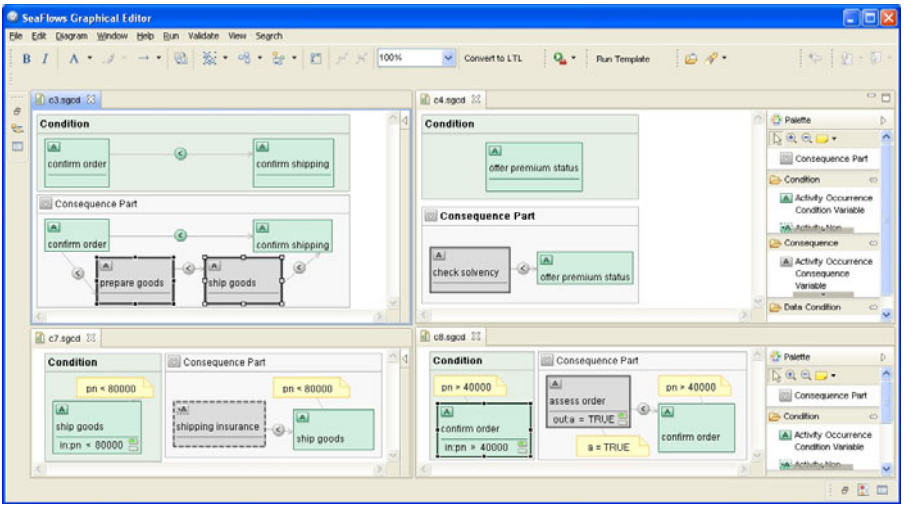

Fig. 9. The SeaFlows Graphical Compliance Rule Editor

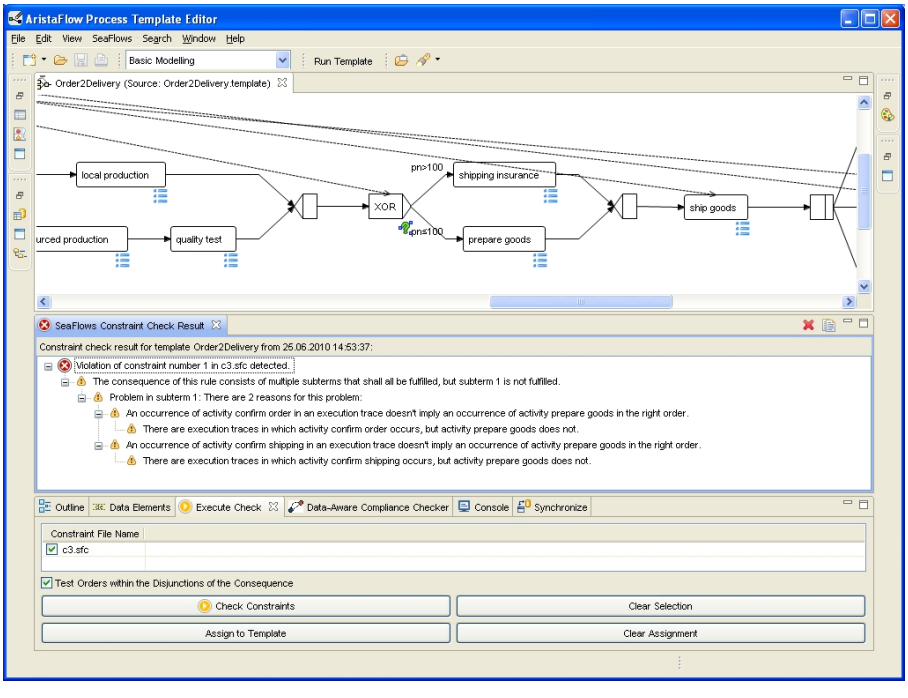

Fig. 10. The SeaFlows Structural Compliance Checker integrated into AristaFlow Process Template Editor

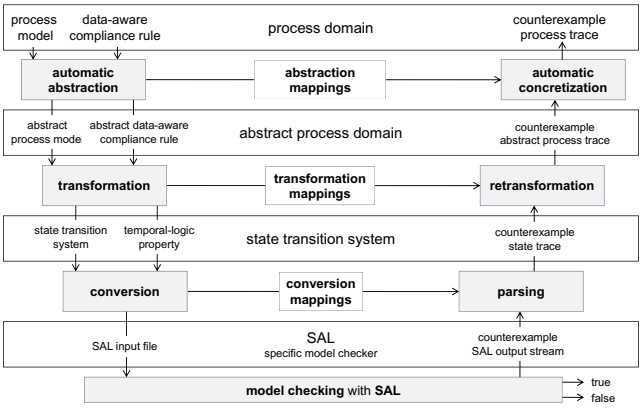

Fig. 11. Data-aware compliance checking and generation of counterexample

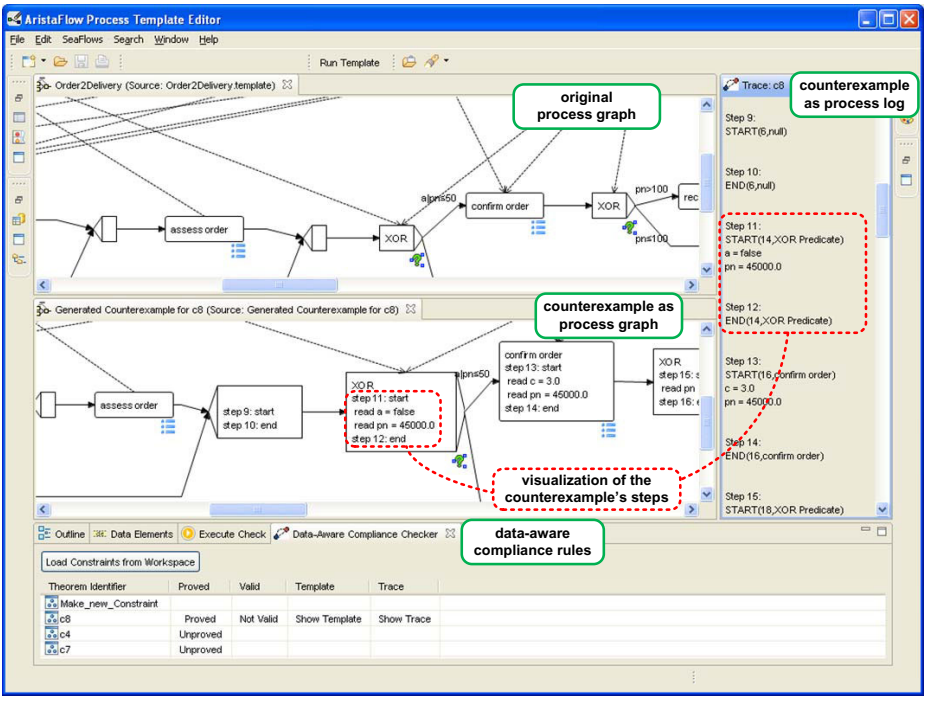

Fig. 12. The Data-aware Compliance Checker visualizes the counterexample as execution trace and process graph

`confirm order` and `confirm shipping` in P. This detailed diagnosis can further be applied to resolve incompliance. For example, the present incompliance can be resolved by repositioning activity `prepare goods` in the P.

The *Structural Compliance Checker* is implemented as an Eclipse-plug-in for the AristaFlow Process Template Editor and thus, is smoothly integrated into the process modeling environment. Therefore, "on the fly" compliance checks during process modeling can be carried out to support compliance by design.

4.3 Data-Aware Compliance Checker

The *Data-aware Compliance Checker* first performs automatic abstraction (cf. Fig. 7). Then, it transforms the abstract process model into a state space representation to apply behavioral compliance checking. For the actual compliance checking, we employed the model checker SAL [12], which, in turn, performs automatic exploration of the state space model and checks for conformance to the compliance rule (cf. Fig. 11). In case incompliance is detected, the *Data-aware Compliance Checker* retransforms the counterexample output of the model checker and visualizes it as an execution trace and within the originial process model. In addition to the violating execution the data condition under which the violation occurred is also illustrated. Fig. 12 shows the output of the checker for the verification of compliance rule c_8 against the order-to-delivery process.

The *Data-aware Compliance Checker* is integrated into the process modeling environment. The class hierarchy comprising about 70 interfaces and 210 classes indicates its complexity. Automatic abstraction is supported for domains of numbers and for large domains of object references.

5 Related Work

Three major challenges arise from compliance verification of process models: compliance rule modeling, verification techniques, and feedback generation. The concepts implemented in SeaFlows Toolset address all three issues. Existing approaches for modeling compliance rules range from rather informal annotations of process models with compliance rules, over formal languages [13], to visual patterns and languages [14, 4, 15]. With the CRGs, we opted for a compositional graph-based modeling formalism that supports the typical antecedent-consequence-structure of rules.

For compliance verification, model checking is often applied in literature [15,14, 13]. As advantage we obtain an approach that is not specific to a particular process meta model or process modeling notation. One challenge of model checking, however, is the generation of meaningful feedback from the report provided by the model checker (e.g., counterexample). The *Data-aware Compliance Checker* addresses this challenge and enables visualization of the counterexample within the process model. The major drawback of model checking is the complexity caused by the necessary model transformations and particularly the exploration of the state space representation. Our structural compliance checking approach exploits specific meta model properties, such as block-structuring, to enable more efficient compliance verification. Some approaches address the verification of data-aware compliance rules [4, 15]. However, the state explosion arising from exploration of the data dimension is not addressed by these approaches. In SeaFlows Toolset we

implemented an abstraction approach that serves as preprocessing step to the actual data-aware compliance checking to limit state explosion.

In [16], Awad et al. address visualization of incompliance by querying the process model for anti-patterns defined for each compliance rule pattern. In our structural approach, structural criteria are automatically derived from CRGs. Checking the structural criteria allows for identifying precisely the structural cause of incompliance.

Similar to DECLARE [17] SeaFlows enables to model graphical compliance rules. In DECLARE constraints are mapped onto formulas in temporal logic and then to finite automata in order to execute constraint-based workflows. In contrast, SeaFlows CRGs are used to verify process models.

SeaFlows Toolset can be further complemented by other process analysis tools, such as the process mining framework ProM [18] to provide comprehensive support of compliance checking a priori as well as a posteriori.

6 Summary and Outlook

In this paper, we introduced concepts underlying SeaFlows Toolset. We showed how compliance rules can be modeled as CRGs to provide a visual process-like and yet still formal representation. In addition, we discussed complementary compliance checking strategies, namely structural compliance checking and behavioral compliance checking. We introduced our structural compliance checking approach based on node relations that enables efficient verification of process models against imposed compliance rules. To address further scenarios, we introduced a behavioral compliance checking approach that addresses data-awareness. SeaFlows Toolset extends process-aware information system by compliance checking functionality. It enables modeling CRGs independently from specific process models by making use of an activity repository. Process models can be enriched by CRGs for documentation purposes and for compliance verification. Two compliance checkers, the *Structural Compliance Checker* and the *Data-aware Compliance Checker*, addressing specific compliance verification scenarios complement each other and thus, ensure broad applicability.

In our future work, we will further extend SeaFlows Toolset to provide support for compliance checking during process execution (cf. the SeaFlows Rule Graph Execution Engine in Fig. 8). In addition, SeaFlows Toolset will be extended by a visualization and explanation component to provide advanced visual user feedback. Finally, case studies can be conducted to validate the concepts implemented in SeaFlows Toolset.

References

1. Sadiq, W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
2. Bobrik, R., Reichert, M., Bauer, T.: View-based process visualization. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 88–95. Springer, Heidelberg (2007)

3. Ly, L.T., Rinderle-Ma, S., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. Information Systems Frontiers, 1–25 (2009)
4. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 500–515. Springer, Heidelberg (2009)
5. Yu, J., Manh, T., Han, J., Jin, Y., Han, Y., Wang, J.: Pattern based property specification and verification for service composition. In: Aberer, K., Peng, Z., Rundensteiner, E.A., Zhang, Y., Li, X. (eds.) WISE 2006. LNCS, vol. 4255, pp. 156–168. Springer, Heidelberg (2006)
6. Namiri, K., Stojanovic, N.: A formal approach for internal controls compliance in business processes. In: 8th Workshop on Business Process Modeling, Development, and Support, BPMDS 2007 (2007)
7. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 9–23. Springer, Heidelberg (2010)
8. Barkaoui, K., Ben Ayed, R., Sbaï, Z.: Workflow soundness verification based on structure theory of petri nets. International Journal of Computing & Information Sciences 5(1), 51–61 (2007)
9. Ly, L.T., Rinderle, S., Dadam, P.: Semantic correctness in adaptive process management systems. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 193–208. Springer, Heidelberg (2006)
10. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 332–346. Springer, Heidelberg (2010)
11. Dadam, P., Reichert, M.: The ADEPT project: a decade of research and development for robust and flexible process support. Computer Science - Research and Development 23(2), 81–97 (2009)
12. Bensalem, S., et al.: An overview of SAL. In: Proc. of the Fifth NASA Langley Formal Methods Workshop, NASA Langley Research Center, pp. 187–196 (2000)
13. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
14. Förster, A., et al.: Verification of business process quality constraints based on visual process patterns. In: Proc. 1st Joint IEEE/IFIP Symposium on Theoretical Aspects of Sofware Engineering, pp. 197–208. IEEE Computer Society, Los Alamitos (2007)
15. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM Systems Journal 46(2), 335–361 (2007)
16. Awad, A., Weske, M.: Visualization of compliance violation in business process models. In: Business Process Management Workshops. LNBIP, vol. 43, pp. 182–193. Springer, Heidelberg (2010)
17. Pesic, M., Schonenberg, M., Sidorova, N., van der Aalst, W.: Constraint-based workflow models: Change made easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
18. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., de Medeiros, A.K.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W(E.), Weijters, A.J.M.M.T.: ProM 4.0: Comprehensive support for *real* process analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)