Business Process Compliance through Reusable Units of Compliant Processes

David Schumm[1], Oktay Turetken[2], Natallia Kokash[3],
Amal Elgammal[2], Frank Leymann[1], and Willem-Jan van den Heuvel[2]

[1] Institute of Architecture of Application Systems (IAAS), University of Stuttgart, Stuttgart, Germany
{Schumm,Leymann}@iaas.uni-stuttgart.de
[2] European Research Institute in Service Science (ERISS), Tilburg University, Tilburg, Netherlands
{o.turetken,a.f.s.a.elgammal,w.j.a.m.vdnheuvel}@uvt.nl
[3] Centrum Wiskunde & Informatica (CWI) Amsterdam, Netherlands
{Natallia.Kokash}@cwi.nl

Abstract. Compliance management is essential for ensuring that organizational business processes and supporting information systems are in accordance with a set of prescribed requirements originating from laws, regulations, and various legislative or technical documents such as Sarbanes-Oxley Act or ISO 17799. As the violation of such requirements may lead to significant punishment for an organization, compliance management should be supported at the very early stages of business process development. In this paper, we present an integrated approach to compliance management that helps process designers to adhere to compliance requirements relevant for their processes. Firstly, we introduce a conceptual model for specifying compliance requirements originating from various compliance sources. Secondly, we propose a framework for augmenting business processes with reusable fragments to ensure process compliance to certain requirements by design. Furthermore, we discuss the formalization of compliance requirements using mathematical logics and integrate the framework for process reuse with automated software verification tools.

Keywords: Compliance, Business Process Management, Process Fragment, Formal Modeling, Process Verification.

1 Introduction

In today's business environment, organizations have to cope with an increasing number of diverse and complex compliance requirements stemming from various laws, regulations, internal or external policies, business contracts etc. This increases the necessity and importance of a comprehensive compliance management solution, which must support compliance throughout all the stages of the business process life cycle. Compliance management ensures that business processes are in accordance with a set of prescribed requirements. It should be considered in three main stages: (i) compliance verification of business process models (static verification at design time), (ii) compliance monitoring of the running instances (dynamic verification at runtime),

F. Daniel and F.M. Facca (Eds.): ICWE 2010 Workshops, LNCS 6385, pp. 325–337, 2010.

and (iii) offline monitoring of the completed business process executions. We consider the static and dynamic verification phases as indispensible and complementary phases for ensuring and managing compliance. This is mainly because offline monitoring is a retrospective approach, which is based on the after-the-fact principle. A preventive focus is fundamentally required in order to achieve sustainability and effectiveness in compliance management.

In this paper we introduce a process-centric approach to compliance management focusing on the design time aspects where reusable units of compliant processes are utilized to augment a process with structures related to compliance. The basic idea is to combine the advantages of compliance checking based on logical formulas with a novel approach for business process reuse. Assume a reusable building block that implements a compliance requirement by means of activities and control dependency among them. We refer to such a building block as a process fragment for compliance, or compliance fragment for short. This fragment can be integrated into an existing process with the intention of making the process compliant to the corresponding compliance requirement. Thus, after the fragment has been integrated into a process, the process should actually comply with the requirement that the fragment implements. However, there is still a possibility that the process design violates the requirements as there is yet no evidence that the fragment has been integrated in the correct manner and in the correct place. The major reasons for an incorrect integration are wrong positioning, wrong concretization, and change of the original fragment design. Therefore, we propose involving rules that represent this compliance requirement in a formal manner. These rules can be checked against the modified process model using advanced methods for process verification to assure compliance.

The steps that have to be performed to provide the assurance of compliant process design are briefly described in the following: at first, a compliance expert defines and formalizes the requirements to which a particular process has to comply with. The resulting formal rules are either associated with existing compliance fragments or with new ones which are developed in cooperation of the compliance expert and a process designer. The compliance fragments which are associated with the rules are then integrated into the process by the process designer. The subsequent verification indicates if all rules could be verified, or if changes on the process are required.

The rest of this paper is organized as follows: In Section 2, we introduce a conceptual model for compliance management on which our work is based. In Section 3, we describe a common industrial scenario, which we use as a running example throughout this paper. Our approach to the development of business process models compliant by design is demonstrated in Section 4. In Section 5, we discuss related work. Finally, in Section 6, we conclude the paper and outline future work.

2 Conceptual Model

Most of the compliance requirements originate from rather generic compliance documents. Compliance requirements may emerge from different sources and can take various forms. They may originate from legislation and regulatory bodies (such as Sarbanes-Oxley and Basel II), standards and code of practices (such as: ISO 9001) and/or business partner contracts. These documents can be ambiguous and thus it is difficult to decide what exactly has to be changed in a business process in order to ensure its

compliance to these requirements. Therefore, an appropriate model for capturing and specifying compliance requirements is needed. In particular, since some parts of such documents may not be relevant for a given process, this model needs to describe compliance requirements and correlate them with business processes that must conform to them. Furthermore, since legislation and regulations tend to change over time, a link to the compliance source should be preserved. The conceptual model depicted in Fig. 1 provides the constructs to manage compliance in business processes.

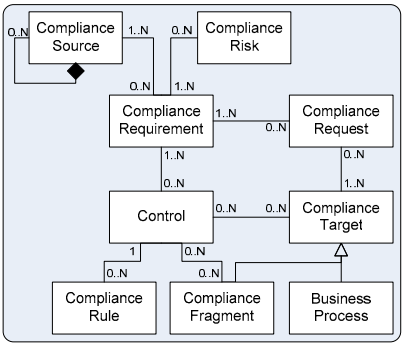

Fig. 1. Conceptual model for compliance management

A *Compliance requirement* is a constraint or assertion that results from the interpretation of the *compliance sources*, such as laws, regulations, policies, standards, contracts, etc. Failure to meet these requirements increases the likelihood of a *compliance risk* to materialize, which in turn might impair the organization's business model, reputation and financial condition. To mitigate these risks and ensure that compliance requirements are satisfied an organization defines *controls*. A control describes the restraining or directing influence to check, verify or enforce rules to satisfy one or more compliance requirements. A *Compliance rule* is an operative definition of a compliance requirement which formally describes a control. A *Compliance fragment* is a connected process structure that can be used as a reusable building block for ensuring a faster and more consistent specification and integration of compliance into a process. Compliance fragments can be used to implement a compliance rule in terms of activities and control structures. A *Compliance target* is a generic specification, such as a business process, or a compliance fragment, which is a target of compliance requirements. A user (compliance or business expert) can issue a *compliance request* to check whether a set of compliance targets conforms to a set of applicable compliance requirements. The purpose of a compliance request is to identify if and how a process can or should be changed to make it (more) compliant.

3 Running Scenario

In order to provide an illustration for the concepts introduced above and to demonstrate our approach we go over a motivating scenario. The general environment in which the scenario takes place is the e-business application domain, and particularly,

banking applications in which compliance to strict regulations and legislations is crucial. Fig. 2 depicts an excerpt from the process model for a "loan origination" process represented using the Business Process Modeling Notation (BPMN). The process starts with the customer submitting a loan request. Once the loan request is received, a credit broker checks if the customer's banking privileges are suspended. Next, a loan threshold is calculated. If the threshold amount is less than 1M Euros, the post processing clerk checks the credit worthiness of the customer through a credit bureau service. If the threshold amount is greater than 1M Euros, the clerk supervisor is responsible for performing the same activities instead of the post processing clerk. Finally, the manager needs to approve the loan form and (in case of acceptance) send the signed form to the customer to sign it.

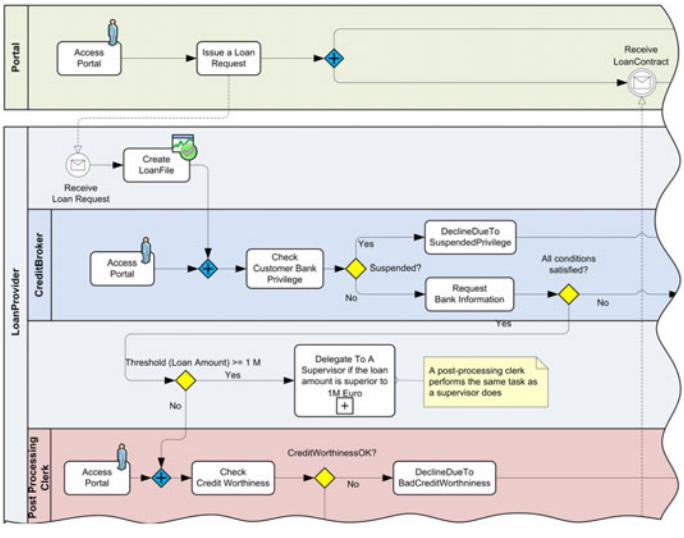

Fig. 2. An excerpt from the BPMN model of the running scenario

There are diverse compliance requirements relevant to this loan origination process, including access rights, temporal aspects, privacy and security. Table 1 gives an example of a compliance requirement regarding the appropriate segregation of duties on the loan origination process. The proposed approach will be discussed by going through this requirement and relevant controls.

Table 1. Compliance requirements relevant for the loan origination process

Control	Compliance Requirement	Comp. Risk	Comp. Source
1- *Customer bank privilege check* is segregated from *credit worthiness check* 2- If the *loan request* exceeds 1M Euros, the *Clerk Supervisor checks the credit worthiness* of the customer 3- The branch office *Manager* checks whether risks are acceptable and makes the final *approval* of the request	Duties in Loan Processing should be adequately segregated	Loan granted with inadequate level of assurance	- Sarbanes-Oxley Sec. 404 - ISO 17799-10.1.3

4 Ensuring Compliance of Business Processes

This section explains how compliance fragment reuse and static process verification can help us to achieve business process compliance by design.

As discussed in the previous section, a process designer is faced with the task of making a process compliant. We assume that an organization has a repository managed by compliance experts where all relevant requirements are stored in a format represented by the aforementioned conceptual model. As a proof of concept, we have implemented such a repository and call it Compliance Requirements Repository (CRR). The designer uses the CRR to find requirements that the particular process needs to adhere to. The 'requirements search' can be a simple keyword search done through all attributes of the requirement (including sources, risks and controls), or be based on an advanced query for expert users.

In response to the designer's request, the CRR returns a list of all relevant requirements. If the discovered requirements have already been instantiated, i.e., formalized as discussed in Section 4.1 and available as concretized process fragments discussed in Section 4.2, they can be directly (re-)used. In this case, the concretized fragments are integrated into the process without the need to check them separately, as their compliant design has been proven before. The augmented process can then be checked against the formal rules by utilizing process verification tools for proving compliant process design (discussed in Section 4.3).

Formal rules can be associated to corresponding compliance requirements with the help of Compliance Request Language Tools (CRLT), discussed in more detail in Section 4.1. If there is one or more abstract fragment that corresponds to a particular rule, it can be concretized and customized by the process designer to fit a specific process. If such a fragment does not exist yet, it can be created and reused in the future. By integrating the fragment into the process, we ensure that the process adheres to the corresponding compliance rules. In our approach, we assume that entities (constructs) present in concrete fragments and compliance rules share unique identifiers in order to provide the correlation.

4.1 Defining and Formalizing Compliance Requirements

Compliance requirement specification language should be based on concepts derived from formal logics to enable automated verification of compliance targets against these requirements. Deontic logic (e.g. [19]) and temporal logic (e.g. [14]) families have been intensively discussed in the literature as a basis for such a specification language. In our framework, we mainly rely on temporal logic for representing compliance rules. Our choice is justified by the fact that system property specification using temporal logics is a mature field supported by efficient verification tools tested and applied in practice for over 20 years. Among the formalisms within temporal logic family, we prefer Linear Temporal Logic (LTL) [16] to Computational Tree Logic (CTL) mainly due to its simplicity, intuitiveness and compositionality of reasoning [22].

One of the main problems of the temporal logic family in general is that logical formulas are difficult to write and understand for users. The notion of *property specification patterns* (Dwyer's property patterns) was introduced in [6] as high-level abstractions of frequently used logical formulas. These patterns assist users in understanding and defining formal specifications. In addition to the original patterns introduced in [6], we have developed *Compliance Patterns* to capture recurring patterns in the compliance context. Table 2 shows such patterns applied to the running scenario. The first control is implemented using the newly introduced *SegregatedFrom* pattern that captures the typical compliance requirement which mandates segregation of duties among different roles and actors. In LTL, *G, F, U* correspond to the temporal operators 'always', 'eventually', and 'until' respectively. *'G'* denotes that formula *f* must be true in all the states of the business process model. '*F'* indicates that formula *f* will be true at some state in the future. '*U'* denotes that if at some state in the future the second formula *g* will be true, then the first formula *f* must be true in all the subsequent states. For example, the LTL representation of 'P *LeadsTo* Q' is '$G(P \rightarrow F(Q))$', which can be read as: If *P* is true, then in the future *Q* should occur.

Table 2. Compliance rules for the examples from the loan origination process

Control	Pattern	Comp. Rules in LTL
1- *Customer bank privilege check* is segregated from *credit worthiness check*	CheckCustomerBankPrivilege *SegregatedFrom* Check Credit Worthiness	*G((CheckCustomerBankPrivilege.Role (Role1) → G(!(CheckCredit Worthiness. Role(Role1))*
2 If the *loan request* exceeds 1M, the *Clerk Supervisor checks the credit worthiness* of the customer	((CreateLoanFile.Threshold >= 1M) *LeadsTo* CheckCredit Worthiness.Role("Supervisor"))	*G((CreateLoanFile.Threshold >= 1M) → F(CheckCredit Worthiness.Role(Supervisor)))*
3- The branch office *Manager* checks whether risks are acceptable and makes the final *approval* of the request.	((JudgeHighRiskLoan AND Approved = "Yes") *Preceeds* SignOfficiallyLoanContract.Role('Manager')) AND ((JudgeHighRiskLoan AND Approved = "No") *Preceeds* DeclineDueToHighRisk('Manager'))	*G((JudgeHighRiskLoan Λ Approved = "Yes") ∨(¬ SignOfficiallyLoanContract.Role('Manager') U (JudgeHighRiskLoan Λ Approved = "Yes"))) Λ G((JudgeHighRiskLoan Λ Approved = "No") ∨(¬ DeclineDueToHighRisk.Role('Manager') U (JudgeHighRiskLoan Λ Approved = "No")))*

We are currently implementing an environment[1] as a part of a tool-suite for business process compliance management. The prototype is a web-based environment, which also incorporates stand-alone tools for building graphical representation of requirements using patterns. The ongoing integration with Process Verification toolkit (see Section 4.3) for process verification is achieved through a group of asynchronous web service calls. BPMN or BPEL representations of compliance targets (i.e. process models) and relevant formal compliance rules specified in LTL are transferred to the Process Verification toolkit. The toolkit returns the verification result, listing the rules that have been checked and whether they are satisfied or not. Fig. 3 presents one of the user interfaces from the implementation reflecting how the results of the compliance check are communicated to the business or compliance expert. The user interface exemplifies the case that the first control given in Table 2 is violated.

[1] CRLT: Compliance Request Language Tools, http://eriss.uvt.nl/compas

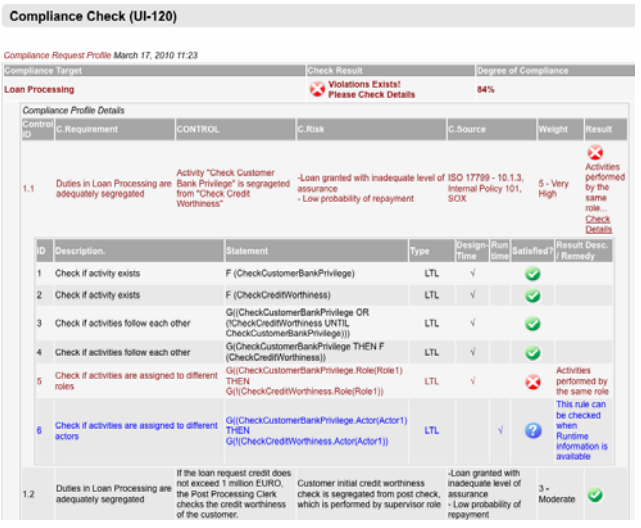

Fig. 3. A user interface with compliance requirements identified for the running scenario

4.2 Compliance Fragments

Process fragments provide a lightweight approach for reusable process structures. In [20] we introduced process fragments for compliance (abbreviated as compliance fragments) as a means to realize compliance requirements within business processes (e.g., based on BPMN) and workflows (e.g., based on BPEL) respectively. In order to utilize this concept for a fast and consistent augmentation of processes with compliance a library of such reusable compliance fragments has to be built up by the bank or a consulting agency in our running scenario. This leads to the first phase in the management life cycle of compliance fragments, which is identification and design. In this phase either reusable process structures related to compliance are identified within an existing process and extracted there from, or they are designed from scratch. For instance, the fragment for approval shown in Fig. 4 could have been extracted from the bank's quality assurance process. To ease reuse the extracted or designed fragment needs to be rendered somewhat abstractly, i.e. static values have to be parameterized, activities need to be generalized and process-specific parts have to be removed (Fig. 4a). A compliance fragment may have multiple points for integration into a process. We call those points fragment entries and fragment exits.

The next phase in the fragment life cycle is storage and retrieval. For this phase we are developing a fragment repository [7] that efficiently supports versioned storage and retrieval. In our example the process designer would query this repository and find (and retrieve) the abstracted fragment for approval. This fragment can then be integrated into the loan approval process in order to realize the compliance requirement. During integration the fragment has to be concretized, i.e. parameters have to be set and the fragment has to be customized for the particular process in which it is applied (see Fig. 4b). Therefore, checking an abstract fragment against concrete rules has little advantages, but it is possible (and useful) to check a concrete rule against a concrete fragment.

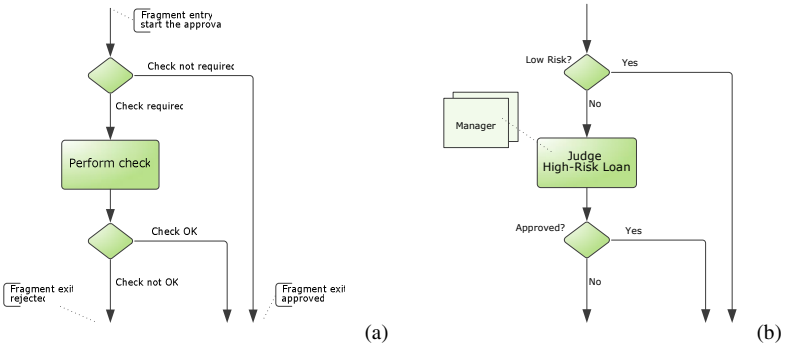

Fig. 4. (a) Abstracted process fragment for approval; (b) Concretized fragment

4.3 Process Verification

To achieve compliance-by-design, we aim at the detection of the violation of compliance rules in design and implementation of compliance fragments and business processes. To accomplish this goal, we automatically convert a compliance fragment or a business process (either in BPMN, BPEL or UML) to its formal representation in Reo [5]. Reo [2] is a graphical channel based coordination language that enables the modeling of complex behavioral protocols using a small set of channel types with predefined behavior. The application of Reo to business process modeling resembles that of Petri nets. Intuitively, an asynchronous FIFO channel with a buffer of capacity one in Reo corresponds to a place in a classical Petri net, while the notion of Petri net transition is generalized and can be composed of multiple synchronous channels. This enables the propagation of synchrony across Reo networks and helps us to model business processes in a more concise and compositional manner. Fig. 5 shows a Reo counterpart for the approval fragment. In this model, an abstract activity *Perform check* is represented as a buffer while conditional gateways correspond to nodes with outgoing filter channels.

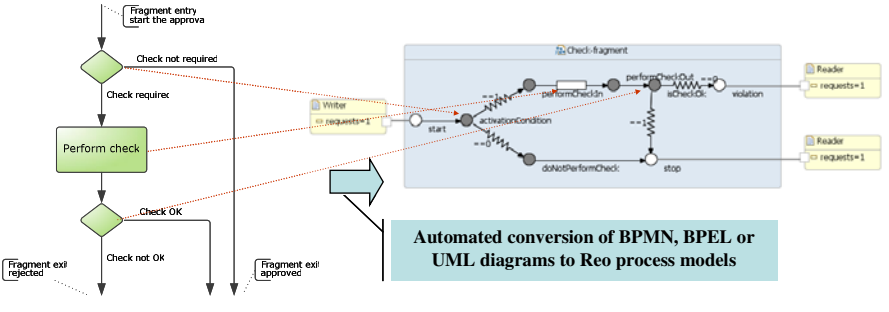

Fig. 5. Process formalization: Abstracted fragment for approval is converted to Reo

Eclipse Coordination Tools (ECT) [3], a supporting framework for behavioral service-based process modeling in Reo, consists of a set of integrated plug-ins that provide the functionality for converting, editing, animating, annotating, simulating and model checking formalized process models. Since high-level models often do not contain all the information necessary for the automated process verification, we assume that ECT is used by a technical specialist to refine the process models that are passed for compliance verification. The imported process models and fragments need to be refined and the compliance rules have to be transformed to a format which can be accepted by a specific model checking tool chosen to verify a given property.

Currently, three model checking tools are supported by ECT, namely Vereofy [23], mCRL2 [15] and PRISM [18]. Vereofy is a tool that can check properties specified in LTL and CTL-like logics and can be used for control flow analysis. Among its advantages are its compatibility with the compliance rule language discussed in Section 4.1 and the ability to visualize counterexamples in a user-friendly manner by showing them on Reo models using flash animations. Detailed examples of using this tool to process compliance analysis, in particular verification of temporal constraints on process control flow and segregation of duties, can be found in [11]. However, data specification supported by Vereofy currently is not elaborate enough to enable the verification of data-dependent compliance rules. Such rules can be analyzed with the help of the mCRL2 toolset. The mCRL2 specification language and the corresponding toolset were developed by the University of Eindhoven and represent a powerful means for large-scale system verification. ECT includes a plug-in for automatic generation of mCRL2 specifications from Reo process models [12], annotated, if necessary, with data and time constraints. For example, Fig. 6 shows the model of the dataflow in a concretized process fragment, where the input data domain is described by a sort *el(activated: Bool, amount: Nat)* which indicates whether the approval is activated and provides the requested loan amount. Data constraints in a format understandable by mCRL2 (e.g., *amount(e1(d)) > 1000000*) are used as annotations to graphical Reo models and specify process dataflow branching conditions.

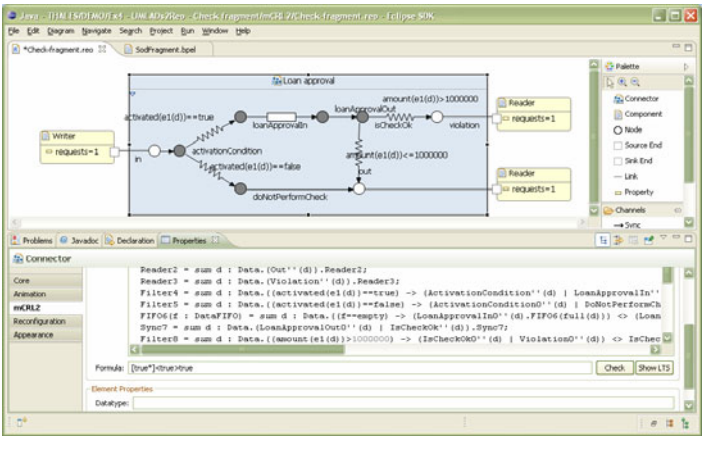

Fig. 6. Formal process model refinement: Concretized fragment for approval is annotated with information about input data domain and dataflow

Such a model can be used for explicit state space generation or model checking against properties specified in a variant of μ-calculus. This format subsumes temporal logics LTL and CTL and allows us to formally express compliance rules with time and data-aware conditions. For example, a compliance rule *"if a requested loan amount is higher than 1M, a manager authorization must be obtained"* corresponds to the following formula:

$$[true^{*}.\exists amount : \mathbf{N}.loanRequest(amount) \wedge (amount > 1000000)]\mu X[\overline{authorization}]X$$

This formula literarily states that for a loan request with the amount exceeding 1M Euro the authorization activity is unavoidable. Finally, the PRISM model checker is used for the verification of probabilistic and quantitative properties of a Reo process model. More detailed study of the application of this tool to compliance analysis constitutes our future work. Apart from process model checking, formalized Reo process models can be used for model-based test generation [21]. In this case, generated tests may assure the compliance of an actual system implementation rather than just the designed model. For example, in the aforementioned scenario at least four test instances should be generated, with and without activated check conditions and loan requests with two amounts: one exceeding 1M, and one not exceeding 1M. Model-based test generation tools such as JTorX are compatible with the generated mCRL2 specifications and can be easily employed in our framework. After the verification, formalized models and model checking results are saved in a repository for further reuse and process reengineering. Counterexamples found by the model checking tools and generated tests that the system did not pass can help the designer to understand why the property violation occurs in the composed process (e.g., detect fragments that are implemented in a wrong way, point out where the wrong integration points or incorrect placements of fragments are).

5 Related Work

Temporal logic has been used intensively in the literature for the formal specification of compliance requirements, key work examples are: [1], [4], [8], [9], [14] and [10]. The authors of [14] proposed a static compliance-checking framework that includes various model transformations. Compliance requirements are modeled using the graphical Business Property Specification Language (BPSL) tool where graphically represented compliance requirements are automatically transformed to LTL formulas. Next, the NuSMV2 model checker is used to verify the compliance. The study in [1] utilized π-Logic to formally represent compliance requirements. In addition, a toolkit has been developed to implement the proposed approach (HAL toolkit).

On the other hand, business process models are abstractly modeled. If the abstract business process model is compliant, a BPEL process equivalent to the abstract representation can be automatically generated. The study in [4] utilized past LTL (PLTL) where properties about the past can be represented. However, sequential compliance requirements are just considered. On the other hand, the study in [16] has utilized the original pattern based system adapted in this paper. They considered only runtime

compliance monitoring though. The study in [10] employed the original pattern specification system used in this paper for the verification of service compositions. In addition, they have introduced the logical composition of patterns using Boolean logical operators. The correctness of pattern composition has also been proved. Composite patterns enable the definition of complex properties in terms of property patterns. Composite patterns can also be used for the specification of complex compliance requirements. Furthermore, authors in [9] have extended the original property pattern system to capture time-related property specifications, so that real-time requirements can be represented via patterns. E.g. activity *A* must always be followed by activity *B* within *k* time units.

Concerning reuse in business processes many concepts have been proposed so far. Besides the well-known approaches for reuse such as sub processes or business rules, more and more lightweight approaches are proposed. For instance, in decentralized process modeling multiple people are involved, each of them having local know-how. Each of the involved designers can model a particular aspect of a process as process fragment, i.e. as an incomplete but connected process structure. These fragments are later composed to a complete process model [13]. Although there is a significant number of works in each of these areas, there is, to the best of our knowledge, currently no approach that combines the advantages of formal languages and compliance checking based on logical formulas with an approach for business process reuse. Here we discussed a concept that demonstrates how these different fields can be combined to support compliance management in business processes.

6 Conclusion and Outlook

In this paper, we presented a framework for design-time business process compliance management. In particular, we introduced a conceptual model for specifying compliance requirements and discussed how these requirements can be stored and processed. The main contribution of this paper is an approach that combines the formalization of compliance requirements, their automated verification for a given process and a novel approach for process reuse. This combination enables a consistent augmentation of business processes with process structures that implement relevant compliance requirements and supports the development of compliant-by-design software applications. By going through a scenario, we briefly demonstrated the concepts and the proposed approach. We also demonstrated the core functionalities of the tools utilized, each representing a part of an ongoing effort on the development of a comprehensive tool-suite for business process compliance management.

Although the formal language introduced in this paper can be used to formalize compliance requirements of diverse types, only those requirements relevant to the control flow of the business processes can be tackled powerfully with compliance fragments. For instance, a locative requirement that demands a certain set of rules on data storage requires a different approach as it refers to database applications rather than activities and control structures. The approach presented in this paper can be seen as one piece of the puzzle in an overall solution to managing compliance.

Acknowledgements

This work is a part of the research project COMPAS (www.compas-ict.eu) which is funded by the European commission, contract no. FP7-215175. Many thanks go to Huy Tran for the development of the process model of the loan origination scenario.

References

1. Abouzaid, F., Mullins, J.: A Calculus for Generation, Verification, and Refinement of BPEL Specifications. In: Proc. of the WWV 2007, pp. 43–68 (2007)
2. Arbab, F.: Reo: A Channel-based Coordination Model for Component Composition. Mathematical Structures in Computer Science 14, 329–366 (2004)
3. Arbab, F., Koehler, C., Maraikar, Z., Moon, Y., Proenca, J.: Modeling, Testing and Executing Reo Connectors with the Eclipse Coordination Tools. In: Tool Demo Session at FACS 2008 (2008)
4. Awad, A., Decker, G., Weske, M.: Efficient Compliance Checking using BPMN-Q and Temporal Logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
5. Changizi, B., Kokash, N., Arbab, F.: A Unified Toolset for Business Process Model Formalization. In: Proc. of the Int. Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA 2010) (2010)
6. Dwyer, M., Avrunin, G., Corbett, J.: Property Specification Patterns for Finite-State Verification. In: Int. Workshop on Formal Methods on Software Practice, pp. 7–15 (1998)
7. Fragmento - Fragment-oriented Repository. Online Documentation (2010), http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm
8. Giblin, C., Liu, A., Muller, S., Pfitzmann, B., Zhou, X.: Regulations Expressed As Logical Models. In: Proc of the 18th Int. Annual Conf. on Legal Knowledge and Information Systems (2005)
9. Gruhn, V., Laue, R.: Specification Patterns for Time-Related Properties. In: 12th Int'l Symposium on Temporal Representation and Reasoning, USA, pp. 198–191 (2005)
10. Yu, J., Manh, T., Han, J., Jin, Y.: 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)
11. Kokash, N., Arbab, F.: Formal Behavioral Modeling and Compliance Analysis for Service-Oriented Systems. In: de Boer, F.S., Bonsangue, M.M., Madelaine, E. (eds.) FMCO 2008. LNCS, vol. 5751, pp. 21–41. Springer, Heidelberg (2009)
12. Kokash, N., Krause, C., de Vink, E.: Data-aware design and verification of service composition with Reo and mCRL2. In: Proc. of the SAC 2010. ACM Press, New York (2010)
13. Eberle, H., Unger, T., Leymann, F.: Process Fragments. In: Proc. of the 17th Int. Conference on Cooperative Information Systems (CoopIS). Springer, Heidelberg (2009)
14. Liu, Y., Muller, S., Xu, K.: A Static Compliance-Checking Framework for Business Process Models. IBM Systems Journal 46 (2007)
15. mCRL2 toolset, http://www.mcrl2.org
16. Namiri, K., Stojanovic, N.: Pattern-based Design and Validation of Business Process Compliance, pp. 59–76. Springer, Heidelberg (2007)
17. Pnueli, A.: The Temporal Logic of Programs, In: Proc. of the 18th IEEE Symposium on Foundations of Computer Science, Providence, pp. 46–57 (1977)

18. Probabilistic model checker, http://www.prismmodelchecker.org/
19. Sadiq, S., Governatori, G., Naimiri, 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)
20. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls. In: Proc. of the MKWI 2010, Universitätsverlag Göttingen (2010)
21. Tretmans, J.: Model Based Testing with Labelled Transition Systems. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST 2008. LNCS, vol. 4949, pp. 1–38. Springer, Heidelberg (2008)
22. Vardi, M.: Branching vs. Linear Time: Final Showdown. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 1–22. Springer, Heidelberg (2001)
23. Vereofy model checking tool, http://www.vereofy.de/